INVERNADERO INTELIGENTE BASADO EN ESP-MESH Y AWS

SMART GREENHOUSE USING ESP-MESH AND AWS

TRABAJO FIN DE MÁSTER
CURSO 2019-2020

AUTOR
MIGUEL ÁNGEL PÉREZ DÍAZ

DIRECTORES
ALBERTO ANTONIO DEL BARRIO
GUILLERMO BOTELLA

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

INVERNADERO INTELIGENTE BASADO EN ESP-MESH Y AWS

SMART GREENHOUSE USING ESP-MESH AND AWS

TRABAJO DE FIN DE MÁSTER EN INTERNET DE LAS COSAS
DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA (DACYA)

AUTOR
MIGUEL ÁNGEL PÉREZ DÍAZ

DIRECTORES
ALBERTO ANTONIO DEL BARRIO
GUILLERMO BOTELLA

CONVOCATORIA: JUNIO 2020
CALIFICACIÓN: 7

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

17 DE JULIO DE 2020

RESUMEN

INVERNADERO INTELIGENTE BASADO EN ESP-MESH Y AWS

El interés por la tecnología de Internet de las cosas se ha extendido de manera significativa durante los últimos años.

Se puede implementar en diferentes campos como son la medicina, industria o agricultura. También es sabido que la utilización de este tipo de dispositivos genera gran cantidad de datos e información. Los actuales servicios disponibles en la nube son de gran utilidad para procesar esta información y definir acciones frente a los valores obtenidos.

El presente trabajo estudia el uso de dispositivos sencillos como son los ESP8266. Estos dispositivos permiten establecer la comunicación con la nube de manera sencilla a través de internet. La información recibida en la nube desde los dispositivos es procesada por diferentes servicios dependiendo de la utilidad. En este estudio se van a obtener valores de temperatura, humedad y luminosidad. Estos datos serán procesados y estudiados en AWS y dependiendo de diferentes reglas establecidas se procederá a notificar al usuario según los umbrales definidos.

Palabras clave

Internet de las cosas, cloud, Amazon Web Services, DynamoDB, función Lambda, ESP8266.

ABSTRACT

Smart greenhouse using ESP-MESH and AWS

Interest in the Internet of Things technology has been extended significantly during last years.

It can be implemented in different fields such as medicine, industry or farming. It is also known that the use of this type of devices generates a large amount of data and information. The current services available in the cloud are very useful for processing this type of information and defining actions based on values.

The present work studies the use of simple devices such as the ESP8266. These devices allow an easy communication with the cloud through the internet. The information received in the cloud from the devices is processed by different services depending on the utility. In this study, values of temperature, humidity and luminosity will be captured. These data will be processed and studied in AWS and depending different rules setup we will proceed to notify the user accordingly to the defined thresholds.

Keywords

Internet of Things, cloud, Amazon Web Services, DynamoDB, Lambda function, ESP8266.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ABREVIATURAS

AWS: Amazon Web Services.

IoT: Internet of Things. Internet de las cosas.

ARN: Amazon Resource Names. Nombres de recursos de Amazon.

SNS: Amazon Simple Notification Service. Servicio de notificaciones de Amazon.

IAM: Identity and Access Management. Administración de identidades.

S3: Simple Cloud Storage Service.

MAC Address: Media Access Control Address.

WLAN: Wireless Local Area Network. Red de área local inalámbrica.

CoAP: Constrained Application Protocol.

DTLS / TLS: Datagram Transport Layer Security / Transport Layer Security

AAA: Authentication, Authorization and Accounting.

KMS: Key Management Service.

CKS: Customer Master Keys.

MQTT: Message Queuing Telemetry Transport.

HTTP: Hypertext Transfer Protocol.

REST: Representational State Transfer. Transferencia de estado representacional.

SOAP: Simple Object Access Protocol.

API: Application Programming Interface. Interfaz de programación de aplicaciones.

MFA: Multi-Factor Authentication. Autenticación multifactor

SDK: Software Development Kit. Kit de desarrollo de software.

IDE: Integrated Development Environment. Entorno de desarrollo integrado.

Capítulo 1 - Introducción

El termino de Internet de las Cosas o también denominado *Internet of Things* en inglés describe los componentes físicos que se comunican a través de redes e interactúan mediante sensores con el entorno. Este tipo de interacciones va a permitir un cambio en la manera en la que las personas desarrollan su actividad diaria.

Internet de las Cosas tiene un amplio espectro de usos como son la medicina, energía, agricultura, hogares y ciudades inteligentes. Se estima un número de dispositivos conectados a internet de más de 20 mil millones en 2020[1].

1.1 Motivación

Se enumeran a continuación cuatro componentes principales que describen la arquitectura y utilidad de Internet de las cosas.

Sistemas empotrados de bajo consumo. Esto hace que el rendimiento se vea incrementado a cambio de un bajo consumo de energía.

Computación en la nube. Haciendo uso de las diferentes plataformas disponibles en la nube como son Microsoft Azure, Google Cloud o Amazon Web Services, se puede recolectar la información de estos dispositivos para poder procesar y almacenar los datos.

Arquitectura de red. Los dispositivos van a ser capaces de comunicarse a través de una red de datos con otros dispositivos, otros sistemas e incluso con la nube a través de Internet.

Análisis de Datos. La gran cantidad de información que pueden recopilar estos dispositivos puede servir para obtener un gran flujo de datos. Esta información se utilizará para una vez procesada obtener resultados en base a datos.

[1] https://www.wired.com/story/wired-guide-internet-of-things/

En este trabajo está enfocado a dos de estos cuatro componentes. La creación de una red de sensores basados en ESP8266 y la utilización del flujo de datos obtenidos de los dispositivos para su almacenamiento y procesado en la nube.

Se utilizan los dispositivos ESP8266 por su tamaño y precio reducidos, así como la facilidad de crear redes distribuidas. Con una conexión a internet a través de un nodo raíz y el acceso a tratamiento de datos en la nube se incrementan las posibilidades de añadir cierta inteligencia al ecosistema que se quiera gestionar.

1.2 Objetivos

- Estudio de los servicios en la nube proporcionados por Amazon Web Services. Recepción de mensajes desde los dispositivos. Procesado de información y actuación frente a eventos específicos.
- Programación de dispositivos ESP8266 con diferentes funciones de sensores y actuadores.
- Creación de un ecosistema de invernadero inteligente.
- Cálculo aproximado del coste de la solución.

1.3 Plan de trabajo

En primer lugar, se pretende estudiar la solución que AWS proporciona para IoT. Desde cuales son los protocolos que se pueden emplear para comunicarse con cloud como los posibles modos de almacenamiento de la información. Para ello se hará uso de la documentación oficial y guías de desarrollo de Amazon Web Services.

También se va a realizar una breve descripción de la arquitectura ESP-MESH que servirá para desplegar de manera masiva dispositivos interconectados entre sí.

Finalmente se desarrollará el caso de uso "**invernadero inteligente**" para lo que será necesario programar los dispositivos ESP8266 con unas características específicas dependiendo de la funcionalidad de cada uno. Dentro de este bloque también se va

a realizar un estudio de diferentes métricas para evaluar el comportamiento de la solución punto a punto.

El cronograma de la figura 1-1 muestra la planificación que se utilizará para realizar el presente trabajo.

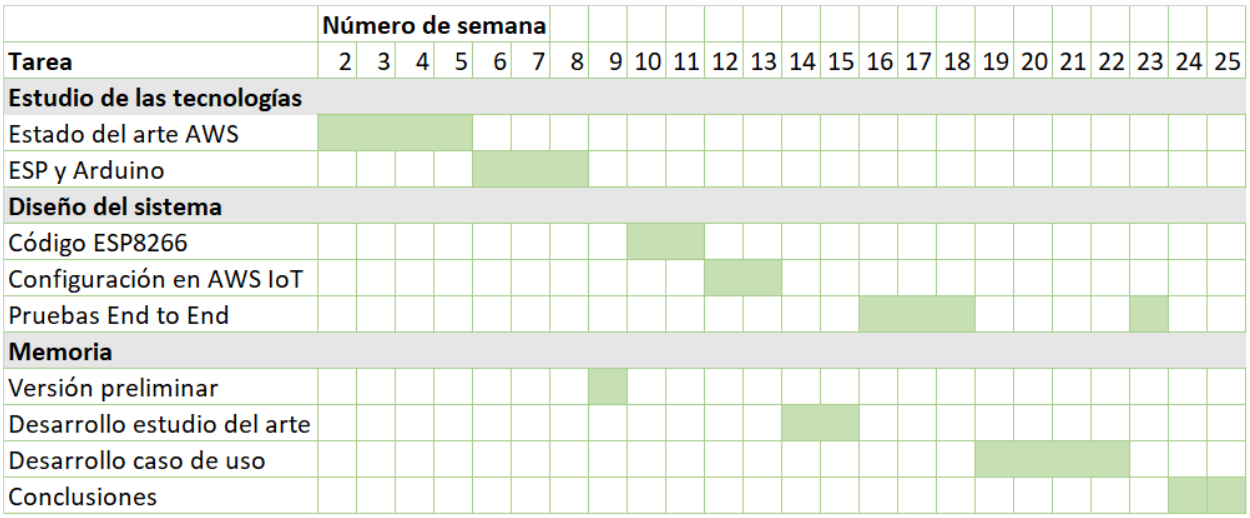


Figura 1-1. Plan de trabajo.

Capítulo 2 - Estado del arte

El concepto de Internet of Things (IoT) se basa en la interacción de pequeños objetos identificados como sensores compuestos de hardware y software. Se utilizan con el fin de recolectar, procesar y distribuir los datos necesarios para la gestión del ecosistema IoT.

Se pueden destacar las siguientes características relacionadas con los elementos que necesita un dispositivo para desarrollar su función.

- Gestión de datos: comunicación, almacenamiento y procesado.
- Comunicación de datos: procesado de datos entre objetos, sensores de datos, hardware.
- Procesado de datos: velocidad, volumen, variedad, variabilidad y veracidad.

Las aplicaciones de IoT en la actualidad son amplias y diversas. Sólo haciendo uso de la movilidad y de la gran cantidad de datos que genera cada usuario a través de sus dispositivos móviles se puede llegar a un proceso de clasificación, análisis y computación muy elevado.

En la Figura 2-1 puede visualizarse el diagrama con los diferentes componentes que pueden llegar a interactuar en un sistema genérico IoT.

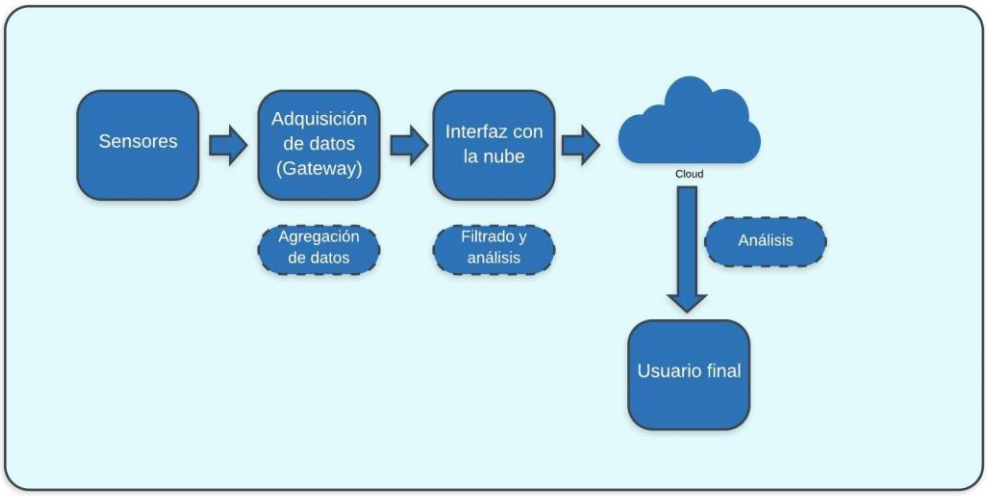


Figura 2-1. Ecosistema IoT genérico.

Se destacan como significativos los siguientes componentes: sensores y actuadores, adquisición de datos, computación en el borde, cloud y usuario final.

AWS IoT proporciona un conjunto de herramientas que permiten conectar dispositivos desplegados en campo con aplicaciones alojadas en la nube. Es posible recopilar información de estos dispositivos para posteriormente almacenarla y procesarla en la nube. Para el estudio de las posibilidades que ofrece AWS en IoT se ha utilizado como referencia la documentación que proporciona AWS para su uso: Guía del desarrollador (Amazon Web Services, AWS IoT - Developer Guide, 2019) además de otras informaciones disponibles en las páginas de documentación de Amazon Web Services.

En este trabajo y dentro del capítulo de estado del arte se van a estudiar los cuatro bloques funcionales en los que se puede dividir el ecosistema de AWS IoT:

- Dispositivos.
- Pasarela IoT.
- Servicios en la nube.
- Aplicaciones IoT.

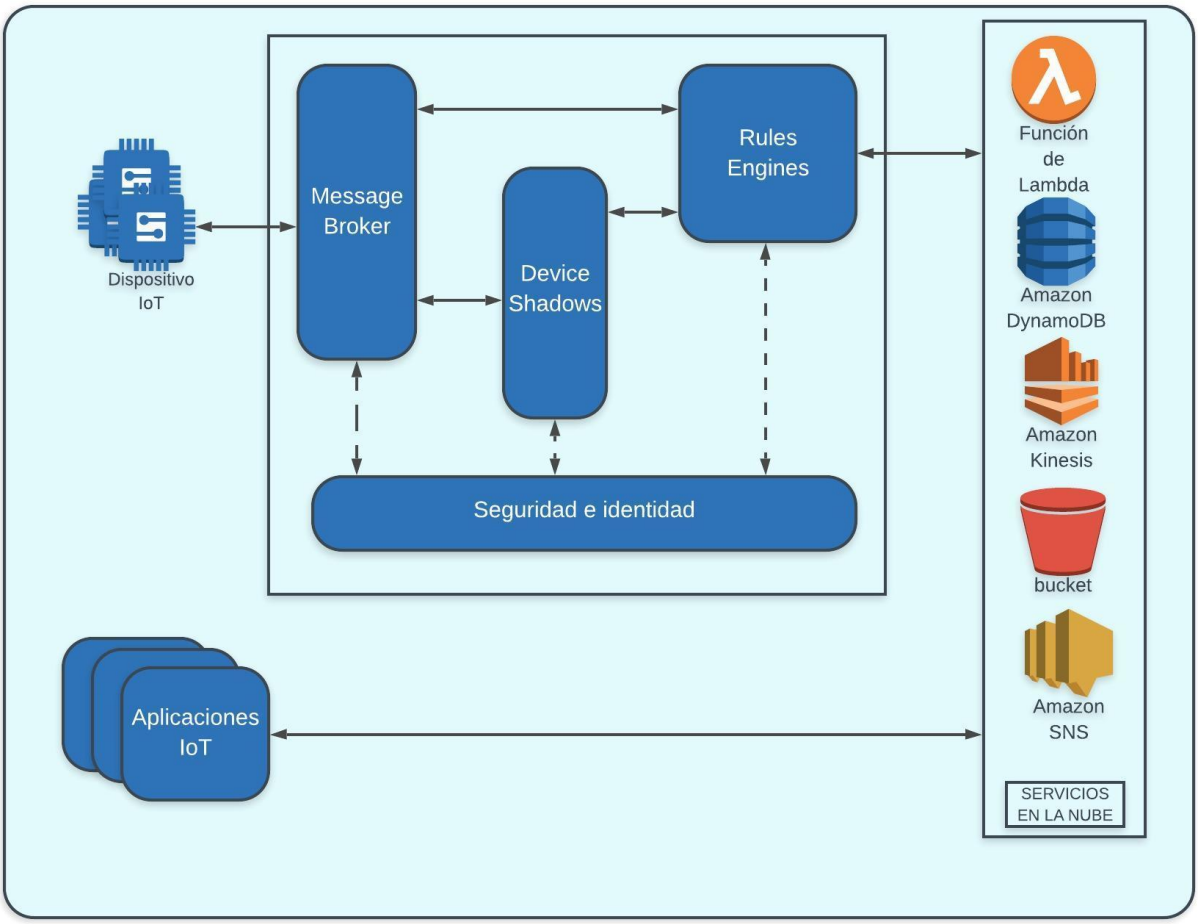


Figura 2-2. Ecosistema AWS IoT.

2.1 Dispositivos

Un dispositivo es un componente que se puede conectar de manera inalámbrica a internet. Su finalidad es la de recopilar información mediante sensores para poder transmitirla y posteriormente en base a su análisis realizar una actuación.

Estos dispositivos conectados van a formar parte de un ecosistema que interactúa con una funcionalidad como puede ser la automatización de un hogar o activación de tareas de elementos industriales.

Los dispositivos conectados para un usuario de consumo personal pueden ser desde televisiones o altavoces inteligentes pasando por sistemas de seguridad comercial o tecnologías inteligentes para ciudades. También se puede encontrar utilidad en monitorización de servicios de regulación de tráfico y semáforos o para actuar en termostatos o aire acondicionado frente condiciones climatológicas.

Como ejemplo de funcionalidad de hogar, así como de uso industrial se describen varios casos de uso:

- Cuando una persona después de aparcar al llegar a casa el coche se comunica con el hogar para transmitirle la orden de activación del termostato a una temperatura preferida. También puede ajustar la intensidad de la luz o incluso modificar su color basándose en los datos obtenidos desde el reloj inteligente con referencia al estado emocional del usuario.

- En el caso de una oficina, los sensores distribuidos en salas de reuniones pueden facilitar al empleado la elección de la sala según el tamaño y recursos disponibles. Una vez están los usuarios dentro de la sala se puede ajustar la temperatura en base a la ocupación y adaptar la luminosidad en base a la interacción de las personas como puede ser la exposición de una presentación en proyector.

- En un nivel más enfocado a la industria productiva, los sensores se pueden utilizar para monitorizar el estado de máquinas de automatización y detectar posibles fallos. En caso de necesidad se puede solicitar un reemplazo de piezas averiadas de manera automática.

Para que estos ejemplos se puedan llevar a cabo es necesaria una gestión efectiva de estos dispositivos. De este modo se puede organizar, integrar, gestionar de manera remota y monitorizar.

Dentro de la gestión de dispositivos se pueden enumerar las siguientes necesidades:

- Registro de dispositivos.
- Autenticación y autorización.
- Configuración remota.

- Primera provisión de dispositivos en campo.
- Monitorización y diagnostico remoto.
- Resolución de incidencias.

Los sensores reaccionan a sensaciones mediante la medida de valores físicos convirtiendo estas medidas en una representación digital. Son capaces de medir un amplio rango de dimensiones proporcionando capacidades sensoriales muy superiores a la del ser humano. Se pueden embeber dentro de objetos físicos conectados de manera sencilla a internet a través de redes cableadas o inalámbricas. Además, son capaces de comunicarse entre sí mediante redes locales de fácil configuración. Un ejemplo de ellas es la red ESP-MESH que pueden generar los dispositivos ESP8266 como veremos posteriormente.

Estas son algunas de las características que definen a los sensores.

- Dependiendo de si producen energía y por tanto necesitan alimentación externa o si sólo reciben energía pueden ser activos o pasivos.
- Si el sensor está integrado en el entorno se considera invasivo.
- En caso de requerir conexión física con el entorno para realizar medidas o no se pueden definir como sensores de contacto o de no-contacto.
- Otra característica que puede distinguir a los sensores es la escala de su medida si es bien absoluta o relativa.
- Los sensores se pueden etiquetar en base a su campo de aplicación. Industria, meteorología, control de transporte son algunos ejemplos.

Tomando como referencia esta última característica, los sensores pueden tener un gran abanico de usos en el campo de la agricultura inteligente. Su utilización puede mejorar de manera significativa la eficiencia, sostenibilidad y rentabilidad de las granjas con un enfoque más tradicional. Algunos ejemplos pueden ser el estudio de la viabilidad de los campos mediante el uso de GPS e imágenes por satélite, el riego o recolección, analíticas en tiempo real e inteligencia artificial.

Una vez obtenidos los parámetros y datos necesarios para proceder a tomar decisiones son los actuadores los que de manera complementaria a los sensores los que reciben información y realizan la función que tienen programada.

De modos similar a los sensores se destacan algunas de las características de los actuadores.

- Se pueden clasificar en base al tipo de movimiento que realizan como por ejemplo lineal, rotatorio o en diferentes ejes.
- Según su emisión de potencia, si es elevada o muy pequeña. También según el tipo de consumo.
- Dependen de los posibles estados de salida.
- En base a su área de aplicación también similar a los sensores: industria transporte, agricultura.

Tanto las características de sensores y actuadores comentadas como información adicional sobre este tipo de dispositivos se pueden obtener en la siguiente referencia [1] (T. Joshva Devatas, 2020).

Los dispositivos se pueden comunicar entre si mediante diferentes protocolos. Entre ellos se encuentran CoAP, DTLS y MQTT. Es este último protocolo el que se va a estudiar en este trabajo ya que Amazon Web Services está integrado con él.

Amazon Web Services permite la gestión de éstos mediante un registro de dispositivos. Utiliza el término "cosas" como la representación de una entidad lógica con una serie de atributos. Un ejemplo sencillo de representación de una cosa en formato puede tener la siguiente estructura:

```
{
	"versión": 3,
	"nombreCosa": "IlumminaciónSalon",
	"idCliente": " IlumminaciónSalon",
	"nombreTipoCosa": "Iluminación",
	"atributos": {
		"modelo": "123",
		"potencia": "75"
	}
}
```

2.1.1 Cosas en AWS IoT

Se puede considerar "cosa" desde un dispositivo físico o sensor hasta una entidad física o lógica, así como una instancia de aplicación.

Dentro de las funcionalidades que ofrece el servicio en la nube de AWS se va a hacer uso del elemento cosa entre otros para poder gestionar los dispositivos que se pueden controlar. Mediante un registro de "cosas" se van a poder representar dispositivos como entidades lógicas.

La información relacionada con los diferentes conceptos de AWS IoT así como su modo de uso descritos a continuación están obtenidas de la documentación oficial de (Amazon Web Services, 2019) referencia [2] de la bibliografía.

Creación de cosa

En general se van a poder utilizar dos maneras de creación de elementos en los servicios de Amazon, la línea de comandos y la consola web. En este capítulo se va a estudiar la manera de definir elementos a través de la línea de comandos ya que también sirve para interpretar la ejecución de estas acciones.

Se puede utilizar el comando ***create-thing*** para crear una cosa.

```
$ aws iot create-thing --thing-name "MyLightBulb" --attribute-payload
"{\"attributes\":
{\"potencia\":\"75\", \"modelo\":\"123\"}}"
```

La tabla 2-1 obtenida de la guía de desarrollo de AWS IoT (Amazon Web Services, 2019) describe los tipos de comandos relacionados con la creación de una cosa en Amazon Web Services.

Tipos de comandos	Estructura
List Things	$ aws iot list-things

Search for Things	$ aws iot describe-thing --thing-name "MyLightBulb"
	$ aws iot list-things --thing-type-name "LightBulb"
	$ aws iot list-things --attribute-name "wattage" --attribute-value "75"

Update a Thing	$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\": {\"wattage\":\"150\", \"model\":\"456\"}}"
Delete a Thing	$ aws iot delete-thing --thing-name "MyThing"
Attach a Principal to a Thing	$ aws iot attach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:useast-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
Detach a Principal from a Thing	$ aws iot detach-thing-principal --thing-name "MyLightBulb" –principal "arn:aws:iot:useast-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"

Tabla 2-1. Comandos de Cosas en AWS.

Tipos de cosas

Los dispositivos que se despliegan en un sistema IoT tienen una función específica tanto para recolección de información como para la ejecución de acciones. Los tipos de cosas van a permitir almacenar información descriptiva y de configuración para todas las cosas asociadas al mismo tipo simplificando la gestión. Definiendo un tipo llamado LightBulb y unos atributos asociados como pueden ser número de serie, fabricante, tipo de dispositivo o voltaje, estos se pueden asociar a las cosas que se vayan a crear en el registro.

Se puede utilizar este comando ***create-thing-type*** para crear un **tipo de cosa**.

```
$ aws iot create-thing-type
--thing-type-name "LightBulb" --thing-type-properties
"thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

Grupos de cosas

Una opción de gestión es la agrupación de dispositivos por grupos. Estos grupos permiten la gestión de varias cosas a la vez mediante una categorización. Es posible también realizar una categorización de grupos de manera jerárquica de modo que asignando una política a un grupo padre, ésta es heredada por sus grupos hijos.

Se puede utilizar el comando ***create-thing-group*** para crear un **grupo de cosas**.

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

2.1.2 Etiquetado de recursos IoT en AWS

El etiquetado en AWS es necesario con el fin de controlar los recursos generados, sus costes asociados o realización de auditorías.

Este etiquetado permite categorizar los recursos de AWS IoT de diferentes maneras. Por tipo de servicio, grupos de dispositivos o departamentos como ejemplo.

Algunas de las sugerencias de AWS para el etiquetado se listan a continuación:

- Utilizar las etiquetas en un formato estándar teniendo en cuenta que diferencia entre mayúsculas y minúsculas. Implementarlo de manera consistente en todos los tipos de recursos.

- Considerar el dimensionamiento de las etiquetas de modo que puedan manejar recursos de control de acceso, automatización, organización y control de coste.

- Si es posible, implementar herramientas automatizadas para la gestión de etiquetas.
- Utilizar el número de etiquetas suficientes sin llegar a complicar la solución.
- Siendo sencillo modificar el etiquetado ya existente para adaptarlo a necesidades del negocio, es necesario controlar las posibles consecuencias tanto para la automatización de despliegues o servicios, así como para el control de acceso o generación de informes de costes.

Tipos de etiquetas

Este es un ejemplo sencillo de cómo se pueden diferenciar etiquetas por tipo. Por convención

Nombre de etiqueta	**Descripción**
Etiquetas técnicas	
Name	Identifica el nombre del recurso
Application_ID	Identifica distintos recursos que utilizan una aplicación común.
Application Role	Describe el rol o función de los recursos: message broker, bbdd.
Environment	Diferencia el tipo de entorno en el que se está trabajando: desarrollo, test, laboratorio, preproducción o producción.
Etiquetas de negocio	
UserID	Responsable de los recursos.
Cost Center	Identifica el centro de coste al que se le puede facturar el recurso.
Customer	Tipo de uso: nombre del cliente o interno.
Project	Sirve para identificar el proyecto al que están asociados los recursos.
Etiquetas de seguridad	
Confidentiality	Grado de confidencialidad de los datos.
Accordance	Se puede utilizar si existen requerimientos de conformidad.

Tabla 2-2. Ejemplos de etiquetas en AWS.

2.2 Pasarela IoT

La pasarela IoT sirve como entrada a la información recibida de dispositivos compatibles con los servicios IoT en AWS. Esta pasarela admite diferentes protocolos de comunicación como MQTT, Web Sockets y HTTP 1.1. En este bloque identificamos varias funcionalidades relacionadas con IoT:

- *Message broker*: Será el encargado de recibir y enviar mensajes entre dispositivos y la nube. Se puede utilizar tanto MQTT como MQTT sobre WebSocket tanto para publicar como para suscribir. También la interface HTTP REST para publicar.

- *Device Shadow / Device Shadow service*: Proporciona información del estado actual de todos los dispositivos de la nube. Funciona en ambas direcciones de modo que se puede actualizar información relativa a un dispositivo para cuando éste se conecte, aunque también puede actualizar su estado el dispositivo.

- *Rules engine*: Encargado de la integración con servicios de AWS. Se puede configurar de modo que extraiga parte de la información del mensaje y la procese o envíe a otros servicios de Amazon como S3 para su almacenamiento, DynamoDB para su inserción en base de datos o una función AWS Lambda para un procesado más complejo.

- *Security and identification*: Va a compartir responsabilidad respecto a la seguridad en la nube con AWS Cloud. El dispositivo mantendrá las credenciales con el fin de poder enviar datos hacia el bróker de manera segura. También el bróker enviará datos de manera segura haciendo uso de las funcionalidades que para ello proporciona AWS.

Estos elementos forman parte del denominado AWS IoT Core. Permiten ingestar, procesar, analizar y almacenar los datos generados por los dispositivos que se conectan a AWS. Algunas de las características de AWS IoT Core son:

- Proporciona autorización, autenticación y encriptación. La autenticación se basa en certificados X.509, encriptación TLS para los datos y políticas IAM (administrador de identidades) para el acceso a recursos y servicios en AWS.

- *Device Gateway* se utiliza para gestionar la conexión de dispositivos permitiendo la recepción y transmisión de mensajes MQTT.

- *AWS IoT Registry* permite el registro de dispositivos. Este registro identifica de manera unívoca a cada dispositivo.

- *Message Broker* gestionará la publicación / suscripción y transmisión de mensajes de manera segura entre dispositivos y aplicaciones.

- *AWS IoT Device SDK* permite tanto la autenticación de dispositivos como el intercambio de mensajes con IoT Core a través de los protocolos MQTT, HTTP, o WebSockets.

- *AWS IoT Rules* facilitará el proceso y análisis de mensajes. Usando un lenguaje basado en SQL se puede seleccionar información específica del mensaje. Después se pueden definir acciones para almacenar información en bases de datos tipo DynamoDB o enviar la información a Kinesis o IoT Analytics.

- *AWS IoT Device* Shadow mantiene un estado de los dispositivos desplegados y conectados a AWS IoT. Este estado es almacenado en un documento JSON utilizado para informar bien del último estado del dispositivo o de su estado futuro. Hará uso de los protocolos MQTT o HTTO para obtener o actualizar información del dispositivo.

2.2.1 Message broker

La función de *Message broker* en MQTT consiste en recibir los datos de los clientes a través de la publicación de un mensaje en un *topic*. Los clientes también reciben los mensajes generados al suscribirse a un *topic*. En AWS *Message broker* realiza esta comunicación entre el sistema IoT de la nube y los clientes.

2.2.2 Protocolos

Message broker utiliza el protocolo MQTT para publicar y suscribirse a *topics* además del protocolo HTTPS para publicar. Ambos protocolos soportan las versiones IPv4 e IPv6. *Message broker* también soporta MQTT sobre el protocolo WebSocket.

PROTOCOLO	**AUTENTICACIÓN**	**PUERTO**	**Nombre APLN**
MQTT	X.509 client certificate	8883, 443	x-amzn-mqtt-ca
HTTPS	X.509 client certificate	8883, 443	x-amzn-mqtt-ca
HTTPS	SigV4	443	N/A
MQTT over WebSocket	SigV4	443	N/A

Tabla 2-3. Protocolos para IoT en AWS.

2.2.2.1 Protocolo MQTT

MQTT es un protocolo de comunicación máquina a máquina que fue desarrollado en 1999 por el Dr. Andy Stanford-Clark (IBM) y por Arlen Nipper (Arcom). Actualmente es un estándar abierto que es mantenido por el grupo OASIS.

Estas son algunas de sus principales características:

- Código de pequeño tamaño.
- Funciona como publicación / subscripción.
- Permite configurar calidad de servicio.
- Existen numerosas librerías disponibles.
- Autenticación mediante usuario y clave con cifrado SSL / TLS.
- Permite la persistencia mediante mensajes almacenados en el *broker*.

Los clientes se dividen entre los que publican mensajes y los que se suscriben a ellos. La comunicación se realiza a través del *broker*.

Un *topic* es una cadena de caracteres que definen una jerarquía que sirve para filtrar los clientes que reciben el mensaje. A continuación, se describe un ejemplo de *topic*.

```
/sensor/nombre_del_nodo/temperatura/nombre_sitio
```

El receptor se puede suscribir tanto a un topic como a un conjunto de ellos utilizando los comodines **#** y **+**.

Message broker permite que los clientes se conecten mediante el protocolo HTTP a través de REST API para poder publicar o suscribirse.

```
<AWS IoT Endpoint>/topics/<url_encoded_topic_name>?qos=1
```

En el desarrollo de caso de uso se describirá más detalladamente el ejemplo de código utilizado para la comunicación, aunque a continuación se muestra un ejemplo de utilización de este protocolo.

```
const char MQTT_HOST[] = "ak6ols07ea57s-ats.iot.eu-west-1.amazonaws.com";
const char MQTT_SUB_TOPIC[] = "device/temperature";
const char MQTT_PUB_TOPIC[] = "device/temperature";
client.publish(MQTT_PUB_TOPIC, shadow, false)
```

2.2.2.2 Device Shadow en AWS

AWS denomina Device Shadow a un documento que sirve para conocer el estado de un dispositivo. Se almacena la información en formato JSON. El dispositivo conectado a internet actualizará esta información mediante los protocolos MQTT o HTTP.

Este servicio actúa como punto intermedio entre aplicaciones y dispositivos permitiendo tanto la obtención de información como la actualización del dispositivo.

De modo similar a un *topic*, el dispositivo puede suscribirse a *device shadow* o realizar actualizaciones. Se pueden enviar actualizaciones desde *shadow* a los suscriptores proporcionando información de la fecha de la última actualización. Las aplicaciones o los dispositivos reciben esta información y en base a los campos desired/reported actúan de una manera u otra.

Después de una actualización, los mensajes son publicados en dos topics MQTT

```
$aws/things/thing-name/shadow/update/accepted
```

Device Shadow envía el mensaje a este *topic* en caso de que la actualización se realice satisfactoriamente. Si no es correcta el mensaje será de tipo /rejected

Si se detecta una diferencia entre la información reportada y la esperada, se envía un mensaje de tipo /delta

Para solicitar la información del último estado almacenado se utiliza /get.

Cuando la petición realizada a Device Shadow finaliza correctamente se envía un mensaje de tipo /accepted. Si es rechazada se envía el mensaje /rejected

```
$aws/things/thing-name/shadow/get/accepted
```

En caso de querer borrar toda la información y contenido del dispositivo en el servicio device shadow se puede utilizar /delete. Para revisar los errores que se producen en el servicio, el cliente se puede suscribir al *topic*: /rejected.

La estructura de un documento de Device Shadow está sujeta a la especificación de un documento JSON. Este sería un ejemplo de la estructura del documento.

```
{
   "state": {
      "desired/reported": {
      }
   },
   "metadata": {
   }
   "version": 1,
   "clientToken" : "UniqueClientToken",
   "timestamp": 1469564492848
}
```

Está compuesto por una serie de campos como son el estado, metadatos, la fecha y hora de documento, token del cliente y versión del documento.

state: desired /reported. Las aplicaciones pueden informar sobre el estado deseado – desired del dispositivo en esta parte del documento. Mientras que con reported es el dispositivo el que informa de un cambio de estado. En este caso las aplicaciones obtienen la información actualizada del estado del dispositivo mediante la lectura del documento.

metadata. Proporciona información de los datos del dispositivo incluidos en el documento. Incluye marca de tiempo del momento en que fue actualizada la información de estado del dispositivo.

timestamp. Informa sobre el momento en el que el mensaje fue transmitido.

clientToken. Cadena de caracteres que identifica unívocamente al dispositivo facilitando la identificación en los mensajes MQTT. Tiene una limitación de 64bytes, generando un error de tipo 400 (Bad Request) y un mensaje de tipo Invalid clientToken.

version. Se genera una nueva versión cada vez que el documento es actualizado. El dispositivo recibe un error si intenta actualizar una versión del dispositivo antigua. De modo similar, si recibe una versión del documento antigua puede decidir cómo actuar.

Frente a un error la respuesta que se va a obtener está basada en una serie de códigos que tienen las siguientes características.

code. Código de respuesta HTTP que indica el tipo de error.

message. Mensaje de texto que proporciona información adicional.

timestamp. Fecha y hora del momento en el que se generó la respuesta.

clientToken. Sólo estará presente si se utilizó para la petición de actualización del documento JSON.

La tabla 2-4 muestra los diferentes códigos de error que se pueden recibir en referencia a operaciones con *Device Shadow*, así como su significado.

Códigos de error	**Mensajes de error**
400 (Bad Request)	JSON no válido (entre otros).
401 (Unauthorized)	No autorizado.
403 (Forbidden)	Prohibido.
404 (Not Found)	"Cosa" no encontrada.
409 (Conflict)	Conflicto de versión.
413 (Payload Too Large)	La carga excede el tamaño máximo permitido.
415 (Unsupported Media Type)	No soportado. La codificación soportada es UTF-8.
429 (Too Many Requests)	Los errores superan el valor de 10.
500 (Internal Server Error)	Fallo interno.

Tabla 2-4. Códigos de error en operaciones Device Shadow.

Es posible actualizar la información de estado a través de la siguiente RESTful API. La información del *endpoint* está asociada a la cuenta que se está utilizando en AWS.

```
https://endpoint/things/thingName/shadow
```

Las opciones que ofrece la RESTful API de *shadow* aparecen descritas en la tabla 2-5.

API	PROTOCOLO	Tipo de petición	Acción
GetThingShadow	HTTP	GET	iot:GetThingShadow
UpdateThingShadow	HTTP	POST	iot:UpdateThingShadow
DeleteThingShadow	HTTP	DELETE	iot:DeleteThingShadow

Tabla 2-5. RESTful API de device shadow.

Los dispositivos pueden utilizar los *topics* MQTT para permitir a aplicaciones y dispositivos obtener, actualizar o borrar información de estado.

La siguiente estructura muestra el formato de la petición MQTT.

```
$aws/things/thingName/shadow/<topic>
```

El ejemplo de Código de uso para la publicación de *topic shadow* MQTT a través de la API es el siguiente.

```
const char MQTT_HOST[] = "ak6ols07ea57s-ats.iot.eu-west-1.amazonaws.com";
const char MQTT_SUB_TOPIC[] = "$aws/things/" THINGNAME "/shadow/update";
const char MQTT_PUB_TOPIC[] = "$aws/things/" THINGNAME "/shadow/update";
client.publish(MQTT_PUB_TOPIC, shadow, false)
```

2.2.2.3 Reglas (rules) en AWS

Las reglas sirven para que los dispositivos interactúen con los servicios disponibles en AWS. Mediante estas reglas se pueden desencadenar diferentes tipos de acciones dependiendo de la configuración.

- Filtrar datos recibidos de un dispositivo.
- Almacenar información en AWS S3.

- Enviar notificaciones a usuarios de tipo *push* mediante SNS.
- Invocar funciones Lambda que ejecuten un código específico.
- Procesar mensajes desde gran número de dispositivos mediante kinesis.
- Obtener métricas y alarmas.
- Enviar información de mensajes para predicciones de Machine Learning.
- Enviar mensajes a Salesforce o canales de analíticas.

La tabla 2-6 muestra algunas de las posibles acciones que se pueden crear como reglas en AWS IoT.

Acciones	**Descripción**
cloudwatchAlarm	Permite modificar un estado de alarma.
cloudwatchMetric	Captura una métrica de Cloudwatch
dynamoDB	Se puede procesar el mensaje MQTT con el fin de almacenarlo en una tabla de la base de datos DynamoDB.
iotAnalytics	Se envían datos del mensaje que utilizó la regla al servicio IoT Analytics para realizar procesados más complejos.
iotEvents	Se envían datos del mensaje al servicio IoT Events facilitando las tares de detección y respuesta a eventos.
kinesis	Permite estribir datos del mensaje MQTT en un flujo Kinesis.
lambda	Realiza una llamada a una función Lambda transfiriendo el mensaje MQTT que activó la regla.
s3	Permite republicar el mensaje a otro topic MQTT.
sns	Envía los datos del mensaje MQTT a una notificación SNS

Tabla 2-6. Posibles acciones basadas en reglas.

Se puede utilizar AWS CLI para realizar la creación de una regla utilizando la siguiente estructura.

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://my-rule.json
```

2.2.3 Seguridad e identificación en AWS

Del mismo modo que la seguridad e identificación es importante en los diferentes tipos de comunicación, también lo es para la transmisión de información en AWS.

Existen diferentes métodos de acceso a AWS IoT dependiendo del tipo de funcionalidad que se necesite utilizar.

- AWS Command Line Interface (AWS CLI). Línea de comandos para la ejecución de tareas en Amazon Web Services.
- AWS IoT API. Gestión de las aplicaciones IoT a través de peticiones HTTP/S.
- AWS SDKs. Creación de aplicaciones para IoT.
- AWS IoT Device SDKs. Creación de aplicaciones para ser ejecutadas en los dispositivos que reciben y envían mensajes.

Para utilizar estos servicios, Amazon Web Services proporciona dos principales enfoques de seguridad en la nube: AAA. Autenticación, autorización y contabilización y la encriptación.

AAA – Authentication, authorization, and accounting

authentication: Identifica al usuario mediante usuario y clave. Adicionalmente se puede utilizar MFA como paso previo al acceso. El servidor de acceso comprobará las credenciales de autenticación con la información almacenada en la base de datos. Si las credenciales coinciden se permitirá el acceso. En caso contrario el acceso es denegado.

authorization: Autorización requerida para ejecutar según qué tareas. Ejecución de ciertos comandos. Las políticas y roles determinarán qué tipo de actividades, recursos o servicios están permitidas para el usuario. Generalmente la autorización se produce en el contexto de la autenticación. Finalizado este, es autorizado para diferentes tipos de accesos y actividades. Esta autorización está basada en roles.

accounting: Mide los recursos consumidos durante el acceso. Puede incluir la cantidad de tiempo logado en el sistema, cantidad de datos utilizados para recepción / transmisión de información. Se lleva a cabo mediante logs de estadísticas de sesión y uso de la información. Su finalidad es el control de autorización, facturación, análisis de tendencias, utilización de recursos actividades de planificación de la capacidad. Utiliza herramientas de AWS como CloudTrail y CloudWatch.

La consola de administración de CloudWatch está disponible la página web de Amazon Web Services.

El siguiente diagrama describe un ejemplo de cuenta en AWS con los correspondientes elementos de recursos y seguridad mencionados. Autenticación y Autorización, Acciones y operaciones en consola y recursos en la nube.

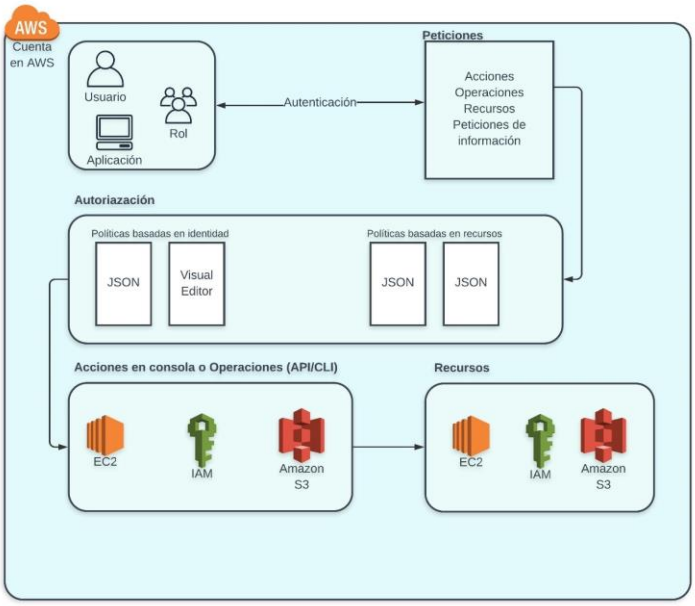


Figura 2-3.Cuentas en AWS.

Se enumeran a continuación varias de las recomendaciones que AWS propone respecto a la seguridad.

- Activar MFA.
- Utilizar restricciones respecto a IPs, Puertos, servicios.
- Eliminar usuarios activos que ya no están presentes en la empresa.
- Otorgar exclusivamente los permisos necesarios.
- Crear grupos por tipos de permisos.
- No utilizar la cuenta ROOT como cuenta de administrador.

Encriptación en AWS

El otro elemento de seguridad que Amazon Web Services proporciona es la encriptación en el lado del servidor que facilita la protección de datos. El servicio de encriptación de AWS *Key Management Service* combina seguridad, alta disponibilidad de hardware y software que proporciona un sistema de gestión de llaves adaptado a la nube. Mediante este sistema se podrán encriptar objetos S3 utilizados en Amazon.

KMS es un servicio que facilita la creación y control de llaves de encriptación. Permite encriptar y desencriptar las llaves de datos (*data keys*) que se utilizan para encriptar datos además está integrado con CloudTrail para la generación de logs.

Las posibles funcionalidades de la encriptación en AWS aparecen descritas a continuación.

- Encriptar / desencriptar datos.
- Generar llaves de encriptación de datos para poder exportar.
- Generar números aleatorios válidos para aplicaciones criptográficas.

Hay dos tipos de llaves, CMK (Customer Master Keys) permiten encriptar y desencriptar hasta 4 kilobytes de datos. *Data Keys* son las llaves de encriptación que se utilizan para encriptar los datos.

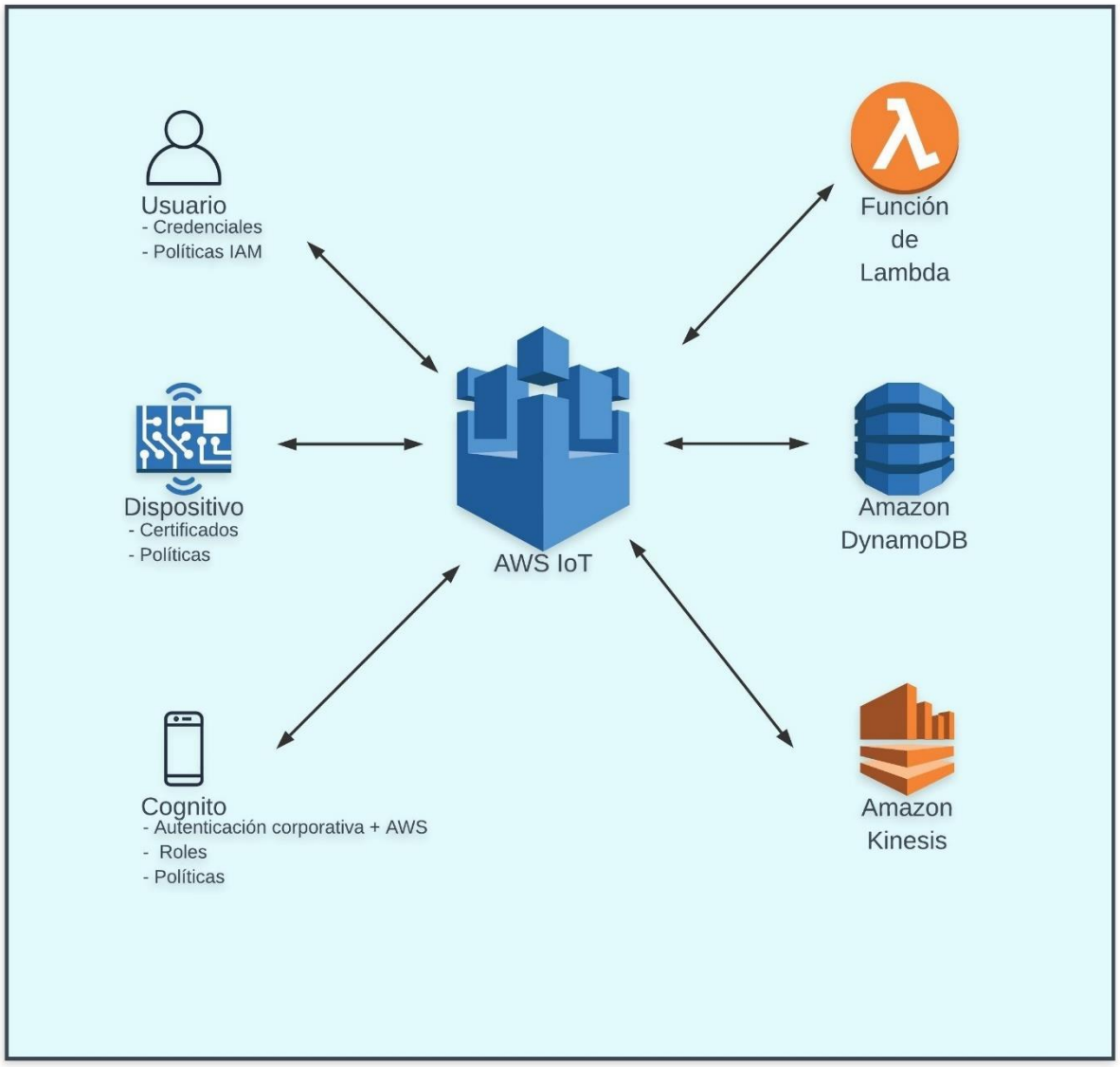


Figura 2-4.Seguridad en AWS IoT.

Estos son los servicios que AWS proporciona para la autenticación en IoT.

Certificados X.509

Son certificados digitales que utilizan el estándar X.509 para asociar una llave pública con una identidad contenida en un certificado. Estos certificados son

proporcionados por una entidad de confianza denominada autoridad de certificación (CA).

Si se quieren utilizar estos certificados IoT de AWS los clientes deben poder utilizar dentro de la implementación TLS:

- TLS1.2
- SHA-256 RSA
- TLS cipher suite

Los dispositivos pueden utilizar los certificados X.509 para autenticarse dentro de los servicios que ofrece AWS IoT. AWS también puede generar certificados mediante la consola, por la línea de comandos utilizando ***create-keys-and-certificate*** o a través de la API *CreateKeysAndCertificate*. Es importante que los dispositivos puedan gestionar la rotación y el reemplazo de certificados para facilitar la operación en caso de expiración de estos.

En el caso del servidor, la autenticación se utiliza para que los dispositivos sepan que se están comunicando con el sistema IoT en AWS y no con otros sistemas que pudieran suplantar este servicio.

Usuarios IAM, grupos y roles

Un **usuario** es una entidad que representa a una persona o servicio. Este usuario permite interactuar con AWS a través de la consola, la API o la línea de comandos CLI.

Asociando este usuario a un **grupo** permite especificar permisos pare este grupo en particular. Estos permisos pueden variar dependiendo de las necesidades: administrador, superusario, operador.

El rol está asociado a una política de permisos que dicho usuario puede realizar. El usuario asumirá un rol determinado dependiendo de las tareas que vaya a realizar. Este rol puede ser temporal o permanente.

Cognito

Amazon Cognito facilita la identificación de usuario a través de otras plataformas proveedoras de identidad como son Amazon, Facebook o Google.

Se pueden asociar diferentes políticas de AWS IoT a Cognito mediante la API AttachPrincipalPolicy.

Autorización

Las posibilidades que un dispositivo puede realizar se basan en políticas. De esta manera podría hasta ejecutar líneas de comando de AWS IoT. Para gestionar estos permisos se utilizan tanto políticas de IAM como específicas de AWS IoT. Se pueden distinguir dos grupos de control:

- Plano de control: utilizado para tareas administrativas como pueden ser crear, actualizar certificados, cosas, reglas y demás elementos. Tipos de políticas en este plano son MQTT sobre autenticación mutua, MQTT sobre WebSocket. HTTP sobre autenticación de servidor, HTTP sobre autenticación mutua.
- Plano de datos: transmisión de datos desde y hacia AWS IoT. En este tipo de plano se utilizarán políticas de tipo HTTP sobre autenticación de servidor.

Estas son las diferentes políticas para AWS IoT.

iot:Connect – Representa el permiso para conectarse al message broker de AWS IoT

iot:Subscribe – Representa el permiso para suscribirse a un topic MQTT

iot:GetThingShadow – Representa el permiso para obtener la información de device shadow del dispositivo.

De una manera más específica se describen las diferentes acciones definidas para AWS IoT en tres bloques: MQTT, recursos shadow y ejecución de trabajos.

Acción	**Descripción**
MQTT	
iot:Connect	Permiso para conectar al message broker. Se comprobará para cada petición de conexión al brker. Por ejemplo, no se permite que dos clientes con el mismo client_id estén conectados a la vez. También permite filtrar clientes no autorizados en base al client_id.
iot:Publish	Permiso para publicar un mensaje MQTT. Se revisa en cada petición.
iot:Receive	Cuando el mensaje se va a entregar se revisa el permiso para un cliente. Se puede utilizar para revocar el permiso en clientes ya suscritos a un topic.
iot:Subscribe	Permiso para suscribirse a un topic. Se revisa en cada petición de suscripción.Es necesario también disponer del permiso iot:Connect.
Shadow	
iot:DeleteThingShadow	Permiso para eliminar un device shadow.
iot:GetThingShadow	Permiso para obtener un device shadow.
iot:UpdateThingShadow	Permiso para actualizar un device shadow.
Job Executions (sólo para HTTPS TLS)	
iot:DescribeJobExecution	Permiso para obtener una ejecución de trabajo para una cosa en particular.
iot:GetPendingJobExecutions	Obtiene la lista de trabajos no finalizados.
iot:UpdateJobExecution	Actualizar una ejecución de trabajo.
iot:StartNextPendingJobExecution	Permiso para comenzar una nueva ejecución de trabajo.

Tabla 2-7. Acciones en AWS IoT para MQTT Shadow y ejecución de trabajos.

2.3 Servicios en la nube

El siguiente bloque funcional dentro del ecosistema AWS IoT es el que facilitan los servicios en la nube. AWS proporciona una serie de servicios con el fin de interactuar con los elementos IoT y facilitar el procesado de información, el almacenamiento de datos y la notificación frente a interacciones.

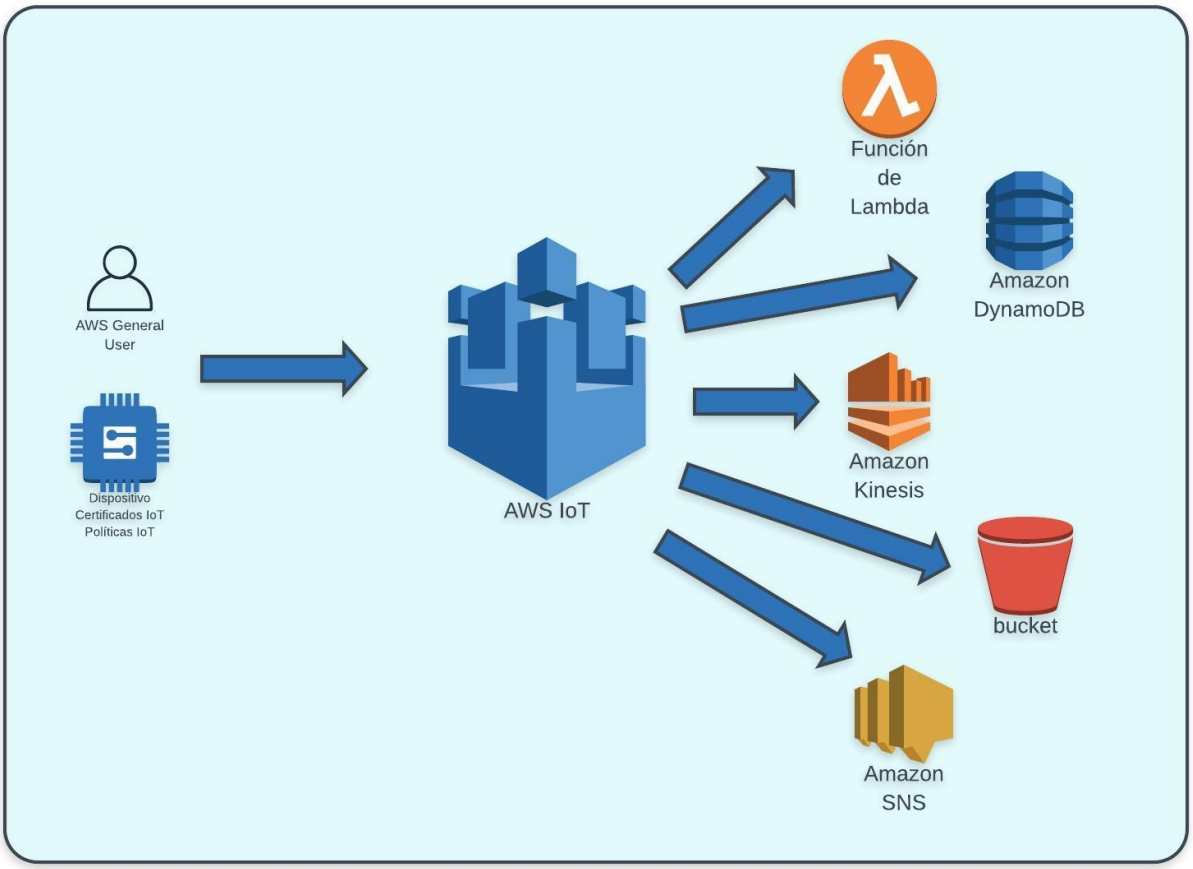


Figura 2-5.Esquema de Amazon IoT Core.

Estos son los principales servicios que se pueden utilizar en la interacción entre dispositivos y la nube.

- Amazon DynamoDB
- Amazon Kinesis
- AWS Lambda
- Amazon S3
- Amazon SNS

2.3.1 DynamoDB

Amazon DynamoDB es una base de datos basada en valor / clave y documentos que proporciona buen rendimiento en cualquier escala. Ofrece diferentes funcionalidades como son multirregión, varios masters en caso de incidencias, backup y recuperación de datos, caché de memoria. Si se necesita ampliar información respecto a DynamoDB se puede tomar como referencia el documento [3] (Amazon Web Services, Amazon DynamoDB. Developer Guide, 2012).

Se destacan tres de las principales funcionalidades de DynamoDB.

- **Rendimiento a escala**. Soporta los dos tipos de modelo mencionados: valor / clave y documentos siendo un esquema flexible. DynamoDB es una base de datos NoSQL que utiliza un número determinado de consultas de manera efectiva. Fuera de esas consultas, el tiempo de respuesta puede incrementarse y su ejecución más costosa. Se buscan las consultas más comunes para que sean lo más rápidas y económicas posible.

- ***Serverless***. La capacidad de los modos de lectura / escritura es proporciona para cada tabla: bajo demanda o provisionado. En caso de tener cargas

de trabajo no planificadas o menos predecibles, se puede utilizar una disponibilidad bajo demanda reduciendo los costes. DynamoDB acomoda estas cagas de trabajo de manera automática según van creciendo o reduciéndose. Se pueden utilizar funciones Lambda para ejecutar funciones que reaccionen a modificaciones de datos en las tablas de DynamoDB.

- **Preparado para producción**. La información se almacena por defecto encriptada. Utilizando AWS Key Management se pueden construir aplicaciones que cumplen los requerimientos regulatorios y de conformidad. Además, facilita el backup y recuperación de datos.

Diseño NoSQL

Estos son los dos conceptos más importantes en relación con una base de datos NoSQL:

- Mantener el menor número de tablas posibles.
- Antes de diseñar el esquema de la base de datos es necesario conocer las necesidades de negocio de las aplicaciones a utilizar, así como los casos de uso.
- Conocer el tamaño y forma de los datos permitirá incrementar la velocidad y facilitará la escalabilidad favoreciendo la efectividad.
- Si se conoce con antelación el pico de carga de consultas también permite anticiparse al uso de la capacidad de entrada / salida. Mantener las consultas de manera distribuida también permite controlar esta capacidad.
- Agrupar los datos relacionados también tiene un impacto en el coste y el rendimiento.

La tabla 2-8 realiza una comparativa entre bases de datos relacionales y NoSQL.

Características	BD relacional	NoSQL (DynamoDB)
Ejemplos de BD	MySQL, Oracle, SQLServer, ProgresSQL.	MongoDB, Redis,ElasticSearch, Cassandra, DynamoDB.
Optimizada para	Consultas específicas, almacenamiento, OLAP.	Aplicaciones Web con escalado como redes sociales, juegos online o IoT.
Modelo de datos	Esquema definido con la información normalizada en tablas, filas y columnas. Relaciones definidas.	Sin esquema. Cada tabla tiene su clave primaria. Información estructurada como ejemplo documentos JSON.
Acceso a la información	Las bases de datos relacionales ofrecen un conjunto de herramientas para acceder a los datos, pero siempre usando SQL.	El acceso a DynamoDB se realiza a través de la consola de gestión de Amazon o mediante AWS CLI. También existen SDKs a disposición de las aplicaciones.
Rendimiento	Las bases de datos relacionales están optimizadas para el almacenamiento de modo que su rendimiento depende del sistema de discos.	Particularmente DynamoDB como servicio gestionado está optimizado para la computación hardware y red.
Escalado	El escalado puede realizarse tanto a hardware más rápido como incrementando el número de hosts en un sistema distribuido.	El escalado en DynamoDB se realiza mediante clusters distribuidos. Es posible especificar el rendimiento sin incrementar la latencia.

Tabla 2-8. Comparativa entre base de datos relacional y NoSQL.

Componentes de la base de datos DynamoDB

Se destacan los principales componentes de una base de datos DynamoDB.

- **Tablas, elementos y atributos.**

La información se almacena en tablas. Las tablas están compuestas de una colección de datos.

Cada tabla estará compuesta de elementos identificable de manera unívoca del resto. En el ejemplo de personas cada elemento está representado por una persona no existe límite en el número de elementos almacenados en una tabla.

Estos elementos se componen a su vez de atributos. Son elementos fundamentales en los datos.

- **Clave primaria.**

Durante la creación de una tabla se especifica tanto su nombre como su clave primaria. Esta clave se utiliza para identificar al elemento de manera unívoca en la tabla. No habrá dos elementos con la misma clave primaria. Existen dos tipos diferentes de clave primaria.

Clave de partición: Es una clave primaria sencilla compuesta de un atributo denominado "clave de partición". Se utiliza como entrada de una función hash. Su salida determina la partición donde se almacena el elemento dentro de DynamoDB.

Clave de partición y clave de clasificación: es una clave primaria compuesta. También utiliza la clave de partición como entrada a la función hash y su salida determina la partición de almacenamiento. El siguiente ejemplo muestra un ejemplo de elemento con clave primaria compuesta (Artista, TituloCancion).

```
{
  "Artista": ElArtista,
  "TituloCancion": "EsElTituloDeLaCancion",
  "TituloAlbum": " EsElTituloDelAlbum",
  "Genero": "pop",
  "Valoracion": 7,
  "AñoPublicacion": "2002"
}
```

- **Índices secundarios.**

Facilitan la búsqueda y consulta de datos en las tablas utilizando una clave alternativa. No es obligatorio el uso de índices, pero permite mayor flexibilidad a las aplicaciones en la consulta de datos. Una vez creados estos índices, se puede consultar la información de manera similar a la de una tabla. Existen dos tipos de índices:

Índice global secundario: Compuesto de clave de partición y de clasificación que pueden ser diferentes a las de la tabla. DynamoDB tiene un límite de 20 índices globales.

Índice local secundario: Utiliza la misma clave de partición que la tabla, pero diferente clave de clasificación. DynamoDB tiene un límite de 20 índices locales.

En el ejemplo anterior de la tabla con Artista como clave de partición, se puede generar un índice que permita consultas utilizando Genero y TituloAlbum.

- **Flujos DynamoDB.**

Es una funcionalidad opcional que captura modificaciones en las tablas de la base de datos. Estas modificaciones van apareciendo en un flujo casi en tiempo real y en el orden en el que ocurren estos eventos. Se produce una escritura en el flujo cuando se produce uno de los siguientes eventos:

- Se añade un nuevo elemento a la tabla.
- Se actualiza un elemento de la tabla.
- Se borra un elemento de la tabla.

Cada entrada del flujo almacena también el nombre de la tabla, la fecha y hora del evento y otros metadatos.

Utilizado en combinación con funciones Lambda se puede activar la ejecución de un código en caso de que aparezca un evento de interés en el flujo.

Distribución de los datos

La forma de almacenar los datos dentro de DynamoDB es mediante particiones. Éstas son espacios de almacén para una tabla localizados en discos de estado sólido SSD y replicados en diferentes zonas de disponibilidad dentro de una región de AWS.

Durante la creación de una tabla, DynamoDB reserva suficiente espacio para almacenar la información. Si el tamaño de la tabla crece se reservarán particiones adicionales para soportar la nueva información.

Según lo indicado anteriormente se utilizará una función hash para calcular la posición de la partición en base a la información introducida en la clave de partición.

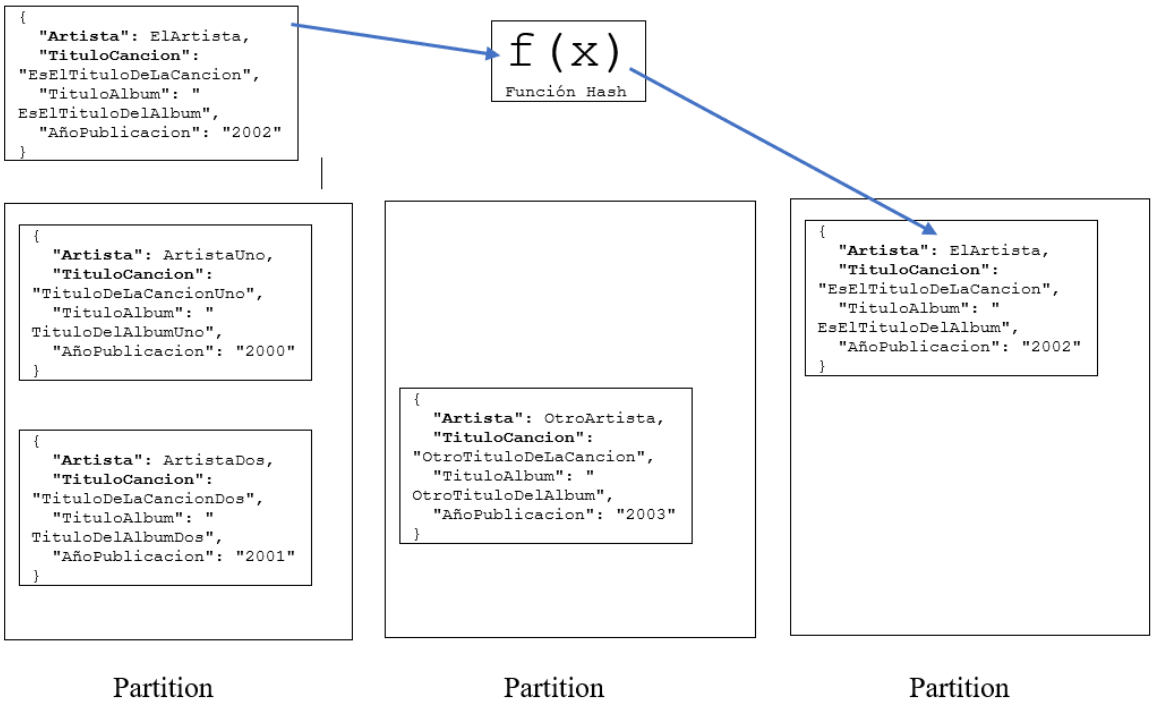


Figura 2-6.Función hash para calcular la posición.

Si es el caso de que se utiliza también la clave de clasificación, además de almacenar la información agrupada con la misma clave de partición, esta información se ordena por el valor de esta clave de clasificación.

Operaciones con tablas

La obtención de información relacionada con una tabla se realiza a través de la operación *DescribeTable*.

```
aws dynamodb describe-table --table-name Music
```

2.3.2 Kinesis

Amazon Kinesis es un servicio gestionado por AWS que se puede utilizar para procesar flujos de datos en tiempo real. Estos datos pueden ser video, audio, datos de telemetría.

Sobre el servicio Kinesis también se destacan las principales funcionalidades:

- Ingesta, control y procesado de datos en tiempo real.
- Auto administrado.
- Escalable para gestionar cualquier volumen de datos.

Estos son los tipos de servicios de Amazon Kinesis:

- Kinesis Video Streams: Registro y procesado de información de video. Transmisión segura, análisis y aprendizaje.
- Kinesis Data Streams: Ingesta y almacenamiento de gran cantidad de datos.
- Kinesis Data Firehose: Almacenamiento de flujo de datos en S3, Redshift o ElasticSearch.
- Kinesis Data Analytics: Análisiss y consultas en tiempo real mediante SQL o Java.

2.3.3 Lambda

AWS Lambda es un servicio que permite ejecutar código sin tener que gestionar la provisión o los servidores que computan. El escalado es automático en caso de necesidad y sólo se utilizan recursos durante el tiempo de computación. Algunos de los lenguajes que puede ejecutar Lambda son: Node.js, Python, Ruby, Java, Go, .NET.

Estas son algunas de las funcionalidades del servicio Lambda:

- Sin necesidad de administrar servidores.
- Auto escalado en función a la carga de trabajo cuando se ejecutan las funciones.
- La facturación se realiza por uso en tramos de 100ms de ejecución de código.

2.3.4 S3

El servicio S3 (Amazon Simple Storage Service) facilita la administración y monitorización del almacenamiento en AWS.

Éstas son algunas de las ventajas que ofrece utilizar el servicio de Amazon S3:

- Creación de buckets para el almacenado de datos. Se puede considerar un bucket como un contenedor dentro de S3 para el almacen de datos.
- Almacenamiento de datos dentro del bucket creado. Cada objeto permite el almacenamiento de hasta 5 TB de datos.
- Descarga de información almacenada.
- Acceso restringido gestionado mediante permisos.
- Interfaces de acceso estándar a través de REST o SOAP.

2.3.5 SNS

El servicio de notificación SNS sirve para la publicación o suscripción de mensajes gestionado autónomamente. Además de poder enviar información a otros servicios como AWS Lambda también se puede utilizar para distribuir notificaciones a usuarios finales a través de envío de correo electrónico y SMS.

Para que el servicio SNS funcione de la manera esperada es necesario crear y controlar un tema específico que será el que haga que los publicadores realicen el envío de mensajes y los suscriptores lo reciban.

Un tema es un punto de acceso lógico que va a actuar como canal de comunicación. Cada tema puede tener uno o varios puntos de envío desde entrada de datos a una función AWS Lambda hasta un enlace HTTP/S o una dirección de correo.

Por tanto, el primer paso de la gestión de SNS es la creación del tema. Para crear la suscripción es necesario seleccionar entre las siguientes opciones.

1) Incluir en la información de suscripción el punto de publicación a través del ARN creado
2) Se elige también el protocolo que va a hacer de punto de publicación, por ejemplo, correo electrónico.
3) Como punto de publicación se especifica la dirección de correo.
4) Por último, se va a crear la suscripción que devolverá un ID si su proceso ha finalizado correctamente.

2.4 Aplicaciones IoT

Se ha hablado de algunas posibles aplicaciones IoT dentro del apartado de dispositivos. Se puede definir la aplicación en entornos IoT desde dos perspectivas. Por una parte, la aplicación o caso de uso punto a punto desde el lado sensores y

dispositivos hasta el usuario final y actuadores. Pero también desde el punto de vista de aplicaciones que hacen uso de AWS IoT para informar y actuar

Para el primer caso se describen brevemente algunos ejemplos

- En el sector sanitario existen diferentes posibilidades de uso tanto para manejar información personal del equipo sanitario de manera privada como para el seguimiento de emergencias sanitarias. El documento [4] (Ekaterina Balandina, 2015)Uno
- Otra aplicación sanitaria también es la de sensores corporales. En este caso se puede realizar un primer análisis de computación en el lado de los sensores y dispositivos y un posterior análisis más completo en la nube. Las conclusiones obtenidas se pueden entregar a una capa final de usuario.

En el segundo caso se pueden incluir las siguientes posibilidades.

- Aplicaciones móviles que representen en formato grafico información proporcionada por ejecuciones de funciones Lambda. El dispositivo móvil necesitará credenciales para realizar las peticiones a AWS siendo el uso de Cognito el preferido.
- Aplicaciones web también pueden hacer uso del API Gateway proporcionado por AWS para la representación gráfica de información.

Capítulo 3 - Arquitectura de un sistema ESP-MESH

3.1 Introducción

El protocolo de red ESP-MESH permite la conexión de gran número de dispositivos distribuidos a lo largo de grandes superficies. Es capaz de construirse y mantenerse de manera autónoma. Utiliza también el protocolo Wi-Fi.

El acceso a internet se realiza a través de un router o Gateway.

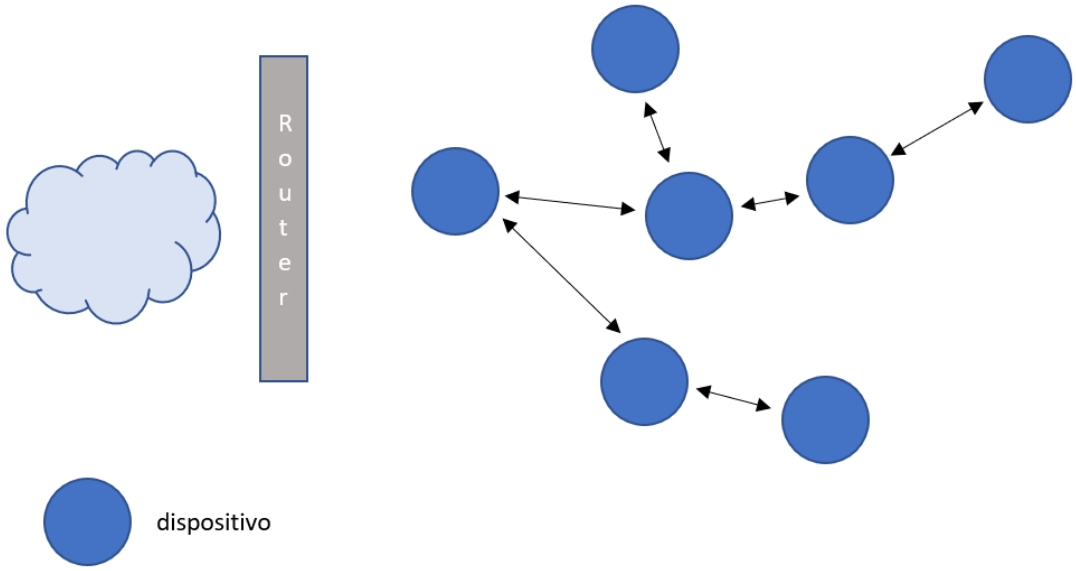


Figura 3-1.Arquitectura de una red ESP-MESH.

La tabla 3-1 resume los términos más comunes de la arquitectura ESP-MESH.

Término	Descripción
Nodo	Dispositivo que forma parte de la red.
Nodo raíz	El nodo principal de la red.
Nodo hijo	Cuando el nodo está conectado con otro nodo en su camino hacia el nodo raíz, este se denomina nodo hijo.
Nodo padre	De manera similar, el nodo que facilita la conexión hacia el nodo raíz se denomina nodo padre.
Conexión	Conexión entre un punto de acceso y un dispositivo. Cada nodo utiliza su interfaz softAP para asociarse con otros nodos y formar la conexión. Este proceso también incluye la autenticación y asociación a una red Wi-Fi. *Upsteam connection*: es la conexión de un nodo hacia su padre. *Downstream connection*: es la conexión de un nodo hacia su hijo.
Salto Wireless	Parte del camino que conecta dos nodos: fuente y destino.
Subred	División entre un nodo y sus nodos descendientes
Dirección MAC	Identifica unívocamente cada dispositivo dentro de la red.

Tabla 3-1. Elementos de una arquitectura ESP-MESH.

3.2 Topología MESH

La arquitectura MESH está basada en varias subredes individuales Wi-Fi que combinadas crean una única WLAN. En este caso los dispositivos funcionan en dos niveles como estación Wi-Fi y punto de acceso. El resultado es una topología de red en forma de árbol con una jerarquía padre-hijo comentada anteriormente.

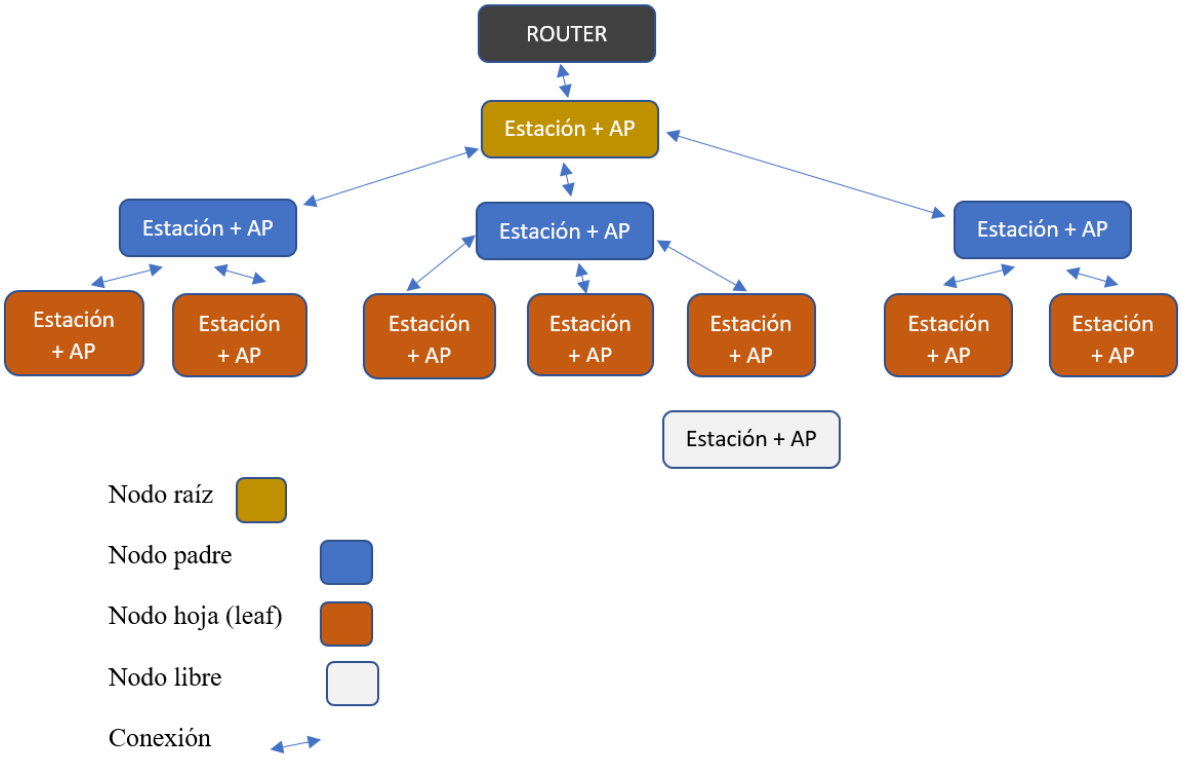


Figura 3-2.Topología de una red ESP-MESH.

En este tipo de topología los nodos transmiten tanto sus propios paquetes como los de otros nodos. En la ilustración anterior se pueden distinguir los siguientes tipos de nodos:

- **Nodo raíz**: Es el nodo de mayor nivel y el que ofrece conectividad con el Gateway. Encamina los paquetes hacia redes externas y recibe también paquetes desde el exterior.

- **Nodo padre**: Los nodos conectados a un nodo hoja o un nodo raíz son considerados nodos padres. Transmiten y reciben paquetes del mismo modo que el nodo raíz.

- **Nodo hoja (leaf):** Es el nodo final dentro de la topología en forma de árbol. No tiene ningún nodo de tipo hijo conectado a él. En este caso sólo transmite o recibe sus propios paquetes. También se pueden asignar a este tipo de nodos los dispositivos que no dispongan del interfaz softAP.

- **Nodo libre**: Son nodos que todavía tienen que establecer una conexión a la red.

- **Nodo padre preferido**. Un nodo libre que tenga que establecer una conexión con la red establecerá esta conexión con su nodo padre preferido en base al siguiente criterio:

  - El nivel en el que el nodo padre candidato está situado

  - El número de conexiones hacia nodos hijos que actualmente tiene establecidas el padre.

En referencia a las tablas de enrutamiento, cada nodo mantendrá una individualmente para realizar el enrutamiento de paquetes correctamente. Cada tabla de enrutamiento en un nodo dispondrá de las direcciones MAC de todos los nodos en esa subred en particular. Las tablas son utilizadas para determinar si los paquetes deben ser redirigidos hacia el nodo padre o hacia el nodo hijo.

Se pueden utilizar diferentes funciones para obtener información de las tablas de enrutamiento.

Función	**Descripción**
esp_mesh_get_routing_table()	Obtiene la tabla de rutas de un nodo.
esp_mesh_get_routing_table_size()	Obtiene el tamaño de la tabla de rutas de un nodo.
esp_mesh_get_subnet_nodes_list()	Obtiene la tabla de rutas de un nodo hijo en particular.
esp_mesh_get_subnet_nodes_num()	Obtiene el tamaño de la tabla de rutas de un nodo hijo.

Tabla 3-2.Información sobre tablas de enrutamiento.

3.3 Proceso de creación de una red ESP-MESH

Como paso previo a la configuración de la red ESP-MESH, parte de la configuración tiene que ser uniforme a lo largo de cada nodo de la red. Para ello los siguientes parámetros deben ser comunes en los nodos.

- Mesh Network ID
- Configuración del Router
- Configuración de softAP

Los pasos de creación de la red se pueden resumir en:

1) En primer lugar, es necesario elegir el nodo raíz que se conectará por una parte al Gateway (router) y por otra parte a sus nodos hijos que forman el camino hacia el resto de los nodos de la topología de árbol.

2) Conectado este nodo al router, los nodos libres dentro del rango de este nodo raíz comienzan a conectarse al nodo raíz formando el segundo nivel de la red. Formada este segundo nivel, los nodos se convierten en padres para continuar con la configuración de los siguientes nodos.

3) El resto de los niveles se forman mediante los nodos que quedan libres convirtiéndose en nodos intermedios o nodos hoja. Este paso se repite hasta que no queden nodos libres.

4) Por último, es necesario limitar el tamaño de la red para prevenir que esta exceda el número máximo permitido de niveles. Los nodos de este nivel límite se convierten en nodos hoja. Los nodos libres que no tienen como referencia ningún nodo padre, quedarán como libres.

3.4 Transmisión de la información

La red ESP-MESH transmitirá paquetes de información con una estructura determinada. Está dividida en la cabecera y los datos.

Cabecera:

- Dirección fuente
- Dirección destino
- Opciones (MESH_OPT_SEND_GROUP, MESH_OPT_RECV_DS_ADDR)

Datos (payload):

- Información de tipo binario o codificada dentro de un protocolo de nivel de aplicación como son HTTP, JSON o MQTT

Estos paquetes a su vez están encapsulados dentro del cuerpo del cuadro de datos Wi-Fi de la siguiente forma:

- Cabecera MAC
- Cuerpo del cuadro
- Secuencia de comprobación

Para realizar una transmisión multicast a un número determinado de nodos se utiliza la siguiente dirección Multicast de grupo como dirección de destino:

```
01:00:5E:xx:xx:xx
```

Esta transmisión multicast permite el envío de paquetes ESP-MES a un grupo de nodos previamente configurado mediante la función ***esp_mesh_set_group_id().***

Si lo que se pretende es enviar la información a todos los nodos de la red entonces se utiliza la funcionalidad Broadcasting. En este caso cada nodo reenvía la información a sus nodos padres e hijos.

Tanto la información recogida en este capítulo como información más amplia sobre la configuración de redes ESP-MESH se puede consultar en la siguiente referencia [5] (espressif, 2020)

Capítulo 4 - Implementación del sistema

4.1 Descripción del caso de uso "invernadero inteligente"

La aproximación a una aplicación real que se va a desarrollar en el trabajo consiste en la gestión autónoma del clima de un invernadero. Mediante sensores de temperatura distribuidos de manera uniforme se puede actuar en momentos determinados del día e incluso en zonas específicas según las necesidades de aclimatación. Desde el caso más sencillo como es la obtención de información de las variables temperatura ambiente y humedad de manera periódica a soluciones más complejas como pueden ser control de temperatura, ventilación, riego o gestión de niveles de dióxido de carbono. Esta información se procesa en AWS y se obtienen parámetros que pueden desencadenar las acciones necesarias.

En este ecosistema de tipo invernadero se podrían implementar respuestas frente a valores obtenidos. Estos son algunos ejemplos de casos prácticos utilizando como dispositivo un sensor de temperatura y humedad.

- Si la temperatura supera un umbral de 35ºC entonces activar el ventilador.
- Si la humedad del suelo desciende por debajo del 70% entonces activar los aspersores hasta que el nivel vuelva a ser de 75% o superior.
- Si la humedad del invernadero desciende por debajo del 80% entonces activar los aspersores hasta que el nivel vuelva a ser de 85% o superior.

Se necesita una distribución homogénea de sensores dentro de la superficie que se pretende controlar. Además, estos sensores necesitarán estar interconectados entre ellos y utilizar una pasarela como salida hacia internet que proporcione la conexión a la nube y a los diferentes servicios de AWS IoT. La figura 4-1 muestra la disposición de dispositivos, su conexión hacia el sistema cloud y la notificación de eventos.

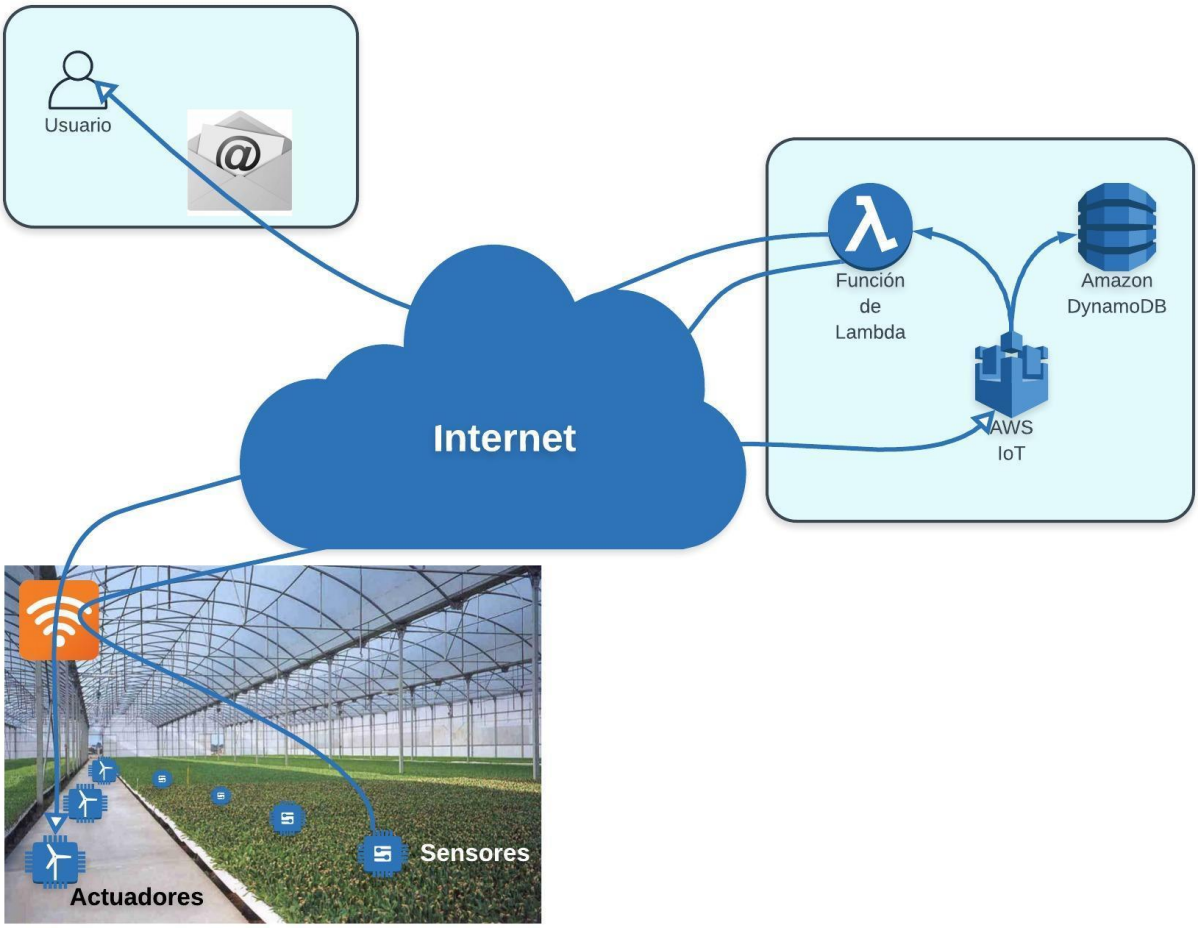


Figura 4-1.Diagrama de conexión invernadero, cloud y usuario final.

La figura 4-2 muestra el diagrama con la arquitectura de la capa de adquisición de datos: las placas, los sensores y actuadores. La distribución de los elementos sensores y actuadores se ha pensado de la siguiente manera.

- Los invernaderos ocupan una superficie de 4 hectáreas distribuidas en cuatro parcelas de 100 metros por 100 metros.
- Se utilizará un sensor de luminosidad para registrar la evolución de la luminosidad a lo largo del día su variabilidad durante las diferentes estaciones.
- Los sensores de temperatura y humedad estarán separados entre sí 25 metros aproximadamente. De este modo en cada parcela constará de 9 dispositivos siendo el total 36.

- Para la activación de los ventiladores se utilizarán 9 dispositivos por parcela distribuidos de manera simétrica.

La comunicación se realiza a través del protocolo MQTT siendo los dispositivos ESP8266 los encargados de interactuar con los sensores y actuadores mediante las conexiones mostradas. Estos dispositivos también hacen de interfaz con la nube a través de una comunicación por red Wi-Fi con la pasarela de acceso a internet.

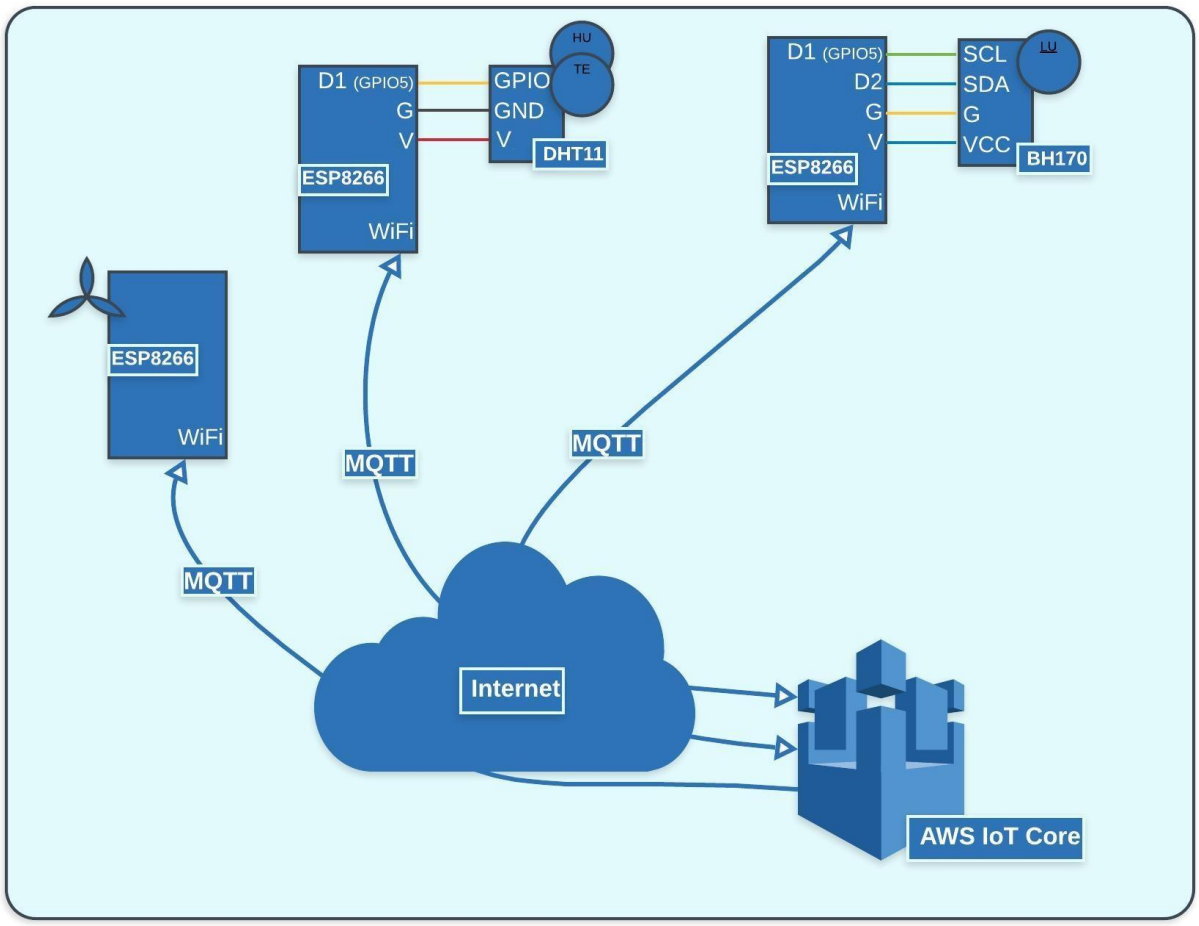


Figura 4-2.Arquitectura de dispositivos ESP8266 y Cloud.

Los mensajes enviados desde un dispositivo a la nube son procesados y analizados. Si un parámetro crítico excede su umbral, se envía una notificación como puede ser el envío de un correo electrónico mediante el servicio SNS al usuario

Para la publicación de datos se utiliza un documento en formato JSON que puede tener la siguiente estructura.

Temperatura y humedad	<pre>{
	  "deviceName":"DHT11",
	  "time":1589913262,
	  "temperature":29.6,
	  "humidity":33
	}</pre>

Luminosidad	<pre>{
	  "deviceName":"BH1750",
	  "time":1591526605,
	  "light":2494
	}</pre>

Ventilador	<pre>{
	  "fan": "0"
	}</pre>

Tabla 4-1. Formato JSON para diferentes medidas.

Se enumeran a continuación los pasos que se ejecutan durante el procesado del mensaje desde que el dispositivo envía el mensaje hasta que el usuario recibe notificación si procede.

1) Publicación del mensaje mediante MQTT hasta el Message Broker utilizando un topic definido: ***device/temperature*** para los datos de temperatura y humedad, ***device/light*** para las medidas de luminosidad y ***device/fan*** para la activación del ventilador una vez superado el umbral de sensación térmica.

2) Procesado de los mensajes recibidos desde *Message Broker* mediante una regla creada (*Rule Engine*).

3) La regla ejecuta una acción en base a la información recibida en el mensaje del dispositivo. Por ejemplo, la ejecución de una función Lambda que analiza los valores actuales del sensor y los compara con respecto al umbral predefinido. La función

Lambda evalúa si el valor límite se ha superado. En caso afirmativo, se envía una notificación de tipo Push mediante SNS al usuario para informar del problema. También se publica el mensaje de activación del ventilador en el *topic* comentado anteriormente.

La figura 4-3 muestra un diagrama de flujos con los elementos que actúan dentro de AWS IoT para la recepción del *topic* ***device/temperatura***. Por una parte, los mensajes recibidos como publicación son insertados en una tabla denominada **IoTESP8266Temperature** de DynamoDB. También se calcula la sensación térmica y se evalúa si el umbral ha sido superado. En caso afirmativo se envía una notificación por correo al usuario.

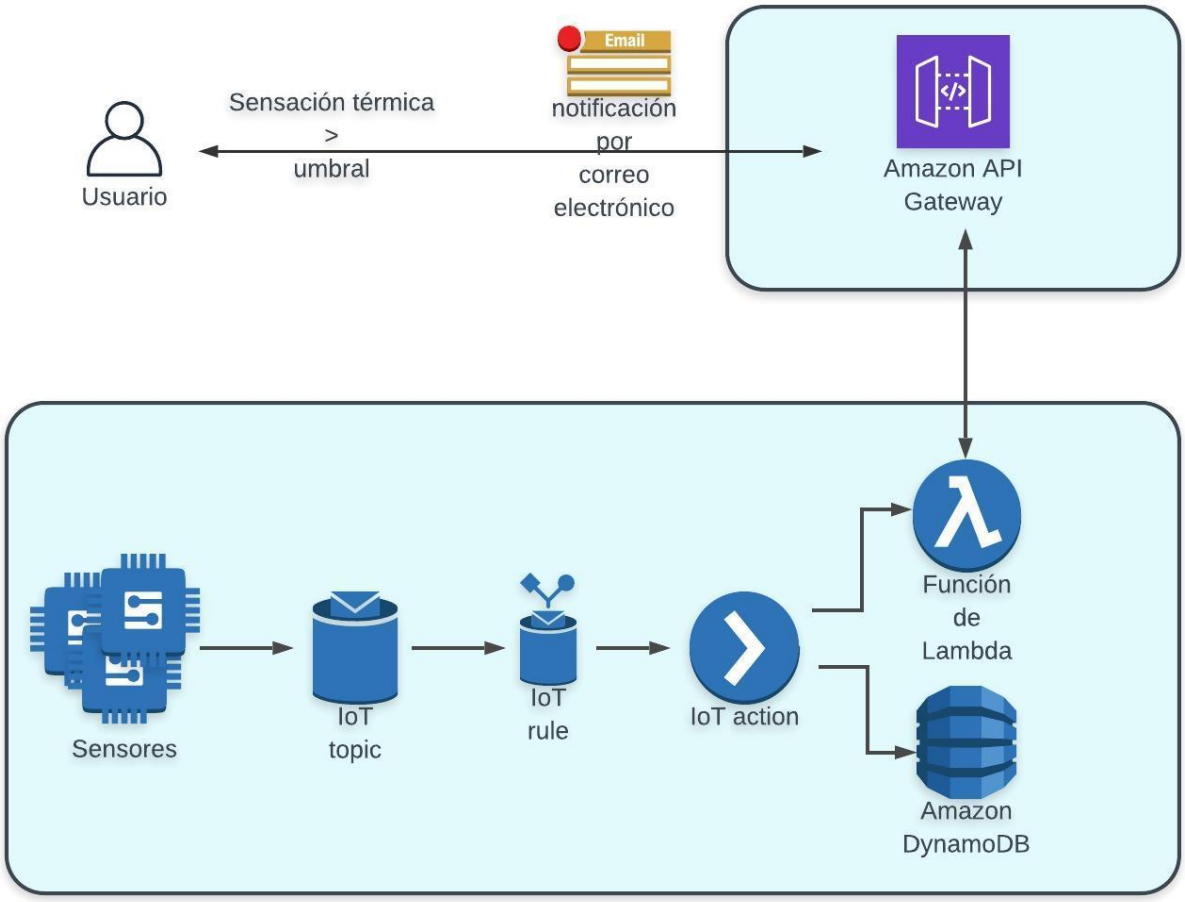


Figura 4-3.Arquitectura del caso de uso IoT.

4.2 Materiales y entorno de desarrollo

Se han comentado varios casos prácticos de reacción frente a determinadas condiciones ambientales. En este apartado se describen los elementos que se han utilizado para llevar a cabo las pruebas de implementación del caso de uso del "invernadero inteligente".

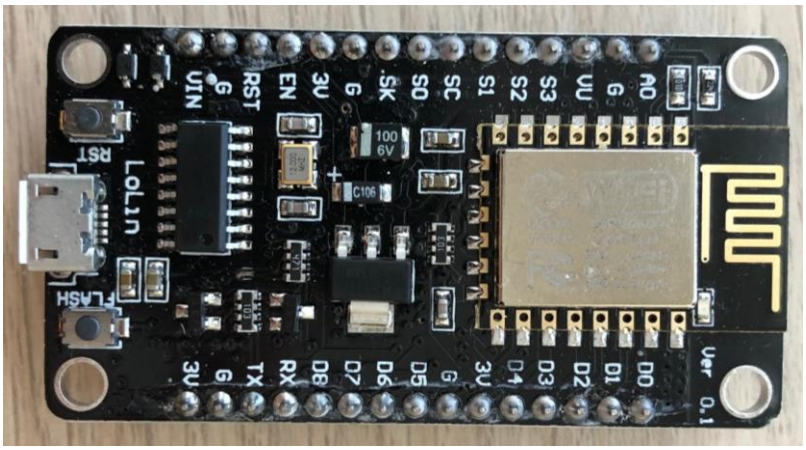
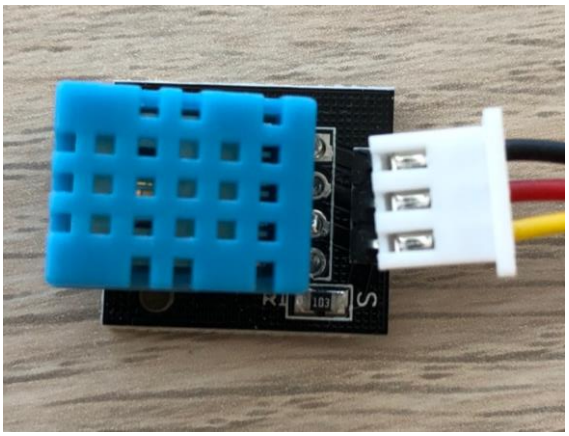
ESP8266	
DHT11	
BH170	

Figura 4-4.Dispositivos, sensores y actuadores.

El entorno de desarrollo utilizado para las pruebas de implementación de la solución está compuesto de los siguientes elementos.

- Ordenador portátil. Intel Core i7 con 3 puertos USB.
  Mediante la aplicación Arduino se ha procedido a programar los dispositivos ESP8266. También se han alimentado estos dispositivos a través de puerto USB y se ha realizado la captura de información a través de la aplicación PuTTY conectada al puerto COM de cada uno.
- 4 dispositivos NodeMCU ESP8266.
  Un dispositivo se ha conectado al sensor de temperatura. El segundo se ha conectado al sensor de luminosidad. El tercer dispositivo se ha utilizado como actuador, al recibir el mensaje "fan" con valor "1" enciende el led.
- 4 cables USB para conectar los dispositivos ESP8266 al ordenador portátil.
- 1 sensor de temperatura y humedad DHT11.
- 2 sensores de luminosidad BH170.
- 11 cables tipo Jumper para conectar los dispositivos ESP8266 con los sensores.
- Acceso a red Wi-Fi.
- Entorno de desarrollo Arduino Software (IDE) 1.8.12.
- Cuenta de AWS para el acceso a través de consola.

La configuración para el entorno de pruebas se ha realizado siguiendo el esquema dibujado en la figura 4-2.

4.3 Creación de la solución "invernadero inteligente"

Para crear el backend en AWS de IoT necesitaremos interactuar con los componentes descritos a continuación.

Dispositivo

- ESP8266 con diferentes funciones de sensores y actuadores.
- Sensores de temperatura / humedad DHT11 y luminosidad BH170.
- Arduino IDE para programar los dispositivos ESP8266.

AWS IoT

- Definición del sistema AWS IoT: cosa, usuario, políticas.
- Creación de funciones Lambda.
- Creación de tablas DynamoDB.
- Configuración de las reglas AWS IoT que permiten la ejecución de acciones.

4.3.1 Configuración del dispositivo ESP8266 mediante Arduino IDE

El dispositivo ESP8266 necesitará conectarse a internet para poder realizar el envío de mensajes MQTT. Esta comunicación se realiza a través del interfaz Wi-Fi del dispositivo. Este es un ejemplo de configuración de red para el dispositivo.

```
#include <ESP8266WiFi.h> //Connect the ESP8266 unit to an existing WiFi access point
#include <PubSubClient.h>

const char* ssid = "+++++++"; //replace this with your WiFi network name
const char* password = "+++++++"; //replace this with your WiFi network password

const char* mqtt_server = "test.mosquitto.org";
```

Para la comunicación MQTT con AWS IoT se puede utilizar un código de tipo publicación suscripción disponibles en repositorios públicos. En este trabajo se ha utilizado el de **debsahu**[2].

El dispositivo ESP8266 debe tener cargados los diferentes certificados en formato DER binario que son necesarios para la comunicación con AWS IoT.

- CA certificate for Amazon AWS.
- certificate.pem.crt.
- private.pem.key.

Se puede utilizar la herramienta **ESP8266 Sketch Data Upload** en Arduino IDE para copiar estos certificados en el dispositivo.

Configuración del sensor ESP8266 como publicador de temperatura y humedad

El dispositivo que se va a utilizar como sensor para obtener las medidas de temperatura y humedad es el modelo DHT11 frecuentemente utilizado como sensor de medida de temperatura. [6] (Iván Manuel Laclaustra, 2016). Estas son algunas de sus características técnicas.

- Medida de la temperatura entre 0ºC y 50ºC con una precisión de 2ºC.
- Medida de la humedad entre 20% y 80% con una precisión del 5%.
- Frecuencia de muestreo de una muestra por segundo.

Para obtener estos datos y que el ESP8266 tenga la función de publicador de temperatura y humedad hay que incluir en el código de Arduino un bloque de configuración del sensor. Los siguientes bloques de código se tienen que incorporar al fichero **PubSubClient.ino**.

[2] https://github.com/debsahu/ESP-MQTT-AWS-IoT-Core/tree/master/Arduino/PubSubClient

```
// DHT CONFIG
#include "DHT.h"
#define DHTTYPE DHT11   // DHT 11
const int DHTPin = 5;     // Pin digital al que conectamos DHT11
DHT dht(DHTPin, DHTTYPE);

const char deviceName[] = "DHT11";
// DHT CONFIG
```

También es necesaria la definición de un bloque para la comunicación con el broker MQTT.

```
//TOPIC CONFIG
const int MQTT_PORT = 8883;
const char MQTT_SUB_TOPIC[] = "device/temperature";
const char MQTT_PUB_TOPIC[] = "device/temperature";
//TOPIC CONFIG
```

Por último, también se define un bloque con la información que va a enviar el dispositivo al *topic* definido previamente.

```
//TOPIC STRUCTURE
  now = time(nullptr);
  state_reported["deviceName"] = deviceName;
  state_reported["time"] = now;
  state_reported["temperature"] = dht.readTemperature();
  state_reported["humidity"] = dht.readHumidity();
//TOPIC STRUCTURE
```

En el fichero **secrets.h** se definen tanto el nombre de la cosa como el punto de enlace de la API REST para comunicar el dispositivo con AWS IoT.

```
#define THINGNAME "IoTESP8266Device"

//punto de enlace de API REST
const char MQTT_HOST[] = "aaaaaaaaaaaaa-ats.iot.eu-west-1.amazonaws.com";
```

Con esta configuración y el sensor DHT conectado al dispositivo ESP8266 éste envía de manera periódica información a la nube.

```
20:34:23.041 -> Sending  [device/temperature]:
{"state":{"reported":{"deviceName":"DHT11","time":1589913262,"temperature":29.6,"humidity":33}}}

20:34:23.144 -> Received [device/temperature]:
{"state":{"reported":{"deviceName":"DHT11","time":1589913262,"temperature":29.6,"humidity":33}}}
```

El dispositivo ESP8266 que ejecuta la función de sensor tiene conectados los siguientes pines al sensor DHT11.

Dispositivo ESP8266	**Sensor DHT11**
VCC	Pin 1 –V
D1 – GPIO5 – válido como entrada/salida y a menudo utilizado como SCL	Pin 2 – GPIO
	Pin 3 – no conectado
G – GND	Pin4 – GND

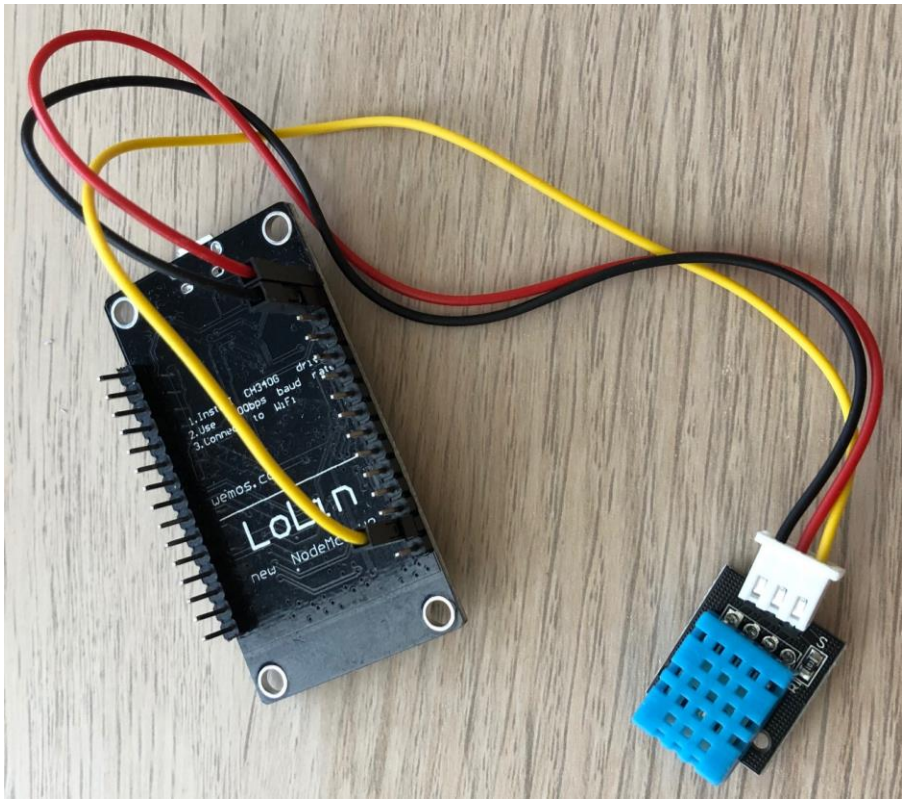


Figura 4-5.Dispositivo ESP8266 y conexión con sensor DHT11.

Configuración del actuador ESP8266 como suscriptor

En el caso del dispositivo configurado como suscriptor / actuador se adapta el código de Arduino para que se suscriba al *topic* que entrega información de actuación *device/fan*. En este caso se va a recibir periódicamente el estado de encendido del ventilador en base a que el valor de sensación térmica supere el umbral.

```
21:14:13.625 -> Received [device/fan]: {"fan": "1"}

21:20:27.184 -> Received [device/fan]: {"fan": "0"}
```

Para la simulación de encendido del ventilador se va a utilizar el LED incluido en ESP8266. Para ello se inicializa GPIO 2 como salida. Y se crea una lógica en el mensaje recibido de modo que si el valor es **"0"** el ventilador se tiene que apagar. Si el valor

recibido en el topic es **"1"** el ventilador se tiene que encender. En el fichero **OnlySubClient.ino** se añade el siguiente código.

```
//// En void setup()
// LED
//Inicializar GPIO 2 como salida
pinMode(LED_BUILTIN, OUTPUT);
// LED


//// En void messageReceived(char *topic, byte *payload, unsigned int length)
  // Se enciende el LED si se recibe un 1 como carácter '1'
  if ((char)payload[9] == '1') {
    digitalWrite(LED_BUILTIN, LOW);   // Encender el LED
  } else if ((char)payload[9] == '0'){
    digitalWrite(LED_BUILTIN, HIGH);  // Apagar el LED
  }
```

En esta ocasión sólo se va a definir el *topic* a que se suscribe el actuador.

```
//TOPIC CONFIG
const int MQTT_PORT = 8883;
const char MQTT_SUB_TOPIC[] = "device/fan";
//TOPIC CONFIG
```

En el fichero **secrets.h** se define el nombre de la cosa, así como el punto de enlace con AWS IoT.

```
////////////////en secrets.h ////////////////////
#define THINGNAME "IoTESP8266Device2"

//punto de enlace de API REST
const char MQTT_HOST[] = "aaaaaaaaaaaaa-ats.iot.eu-west-1.amazonaws.com";
```

Configuración del sensor ESP8266 como publicador de luminosidad

Con el fin de poder obtener la variación de luminosidad durante el día y así obtener una hora óptima para el riego se va a desplegar un sensor de luminosidad que proporcione dicho valor medido en lux con muestras horarias. Se va a utilizar el sensor BH1750 y éste es el código de Arduino necesario incorporar para esta comunicación en el fichero **luzBH1750.ino**.

```
// BH1750 CONFIG

#include <Wire.h>
#include <BH1750.h>


BH1750 Luxometro(0x23);
const char deviceName[] = "BH1750";

// BH1750 CONFIG
```

En el siguiente bloque se define la comunicación con el broker MQTT.

```
//TOPIC CONFIG
const int MQTT_PORT = 8883;
const char MQTT_SUB_TOPIC[] = "device/light";
const char MQTT_PUB_TOPIC[] = "device/light";
//TOPIC CONFIG
```

También se define un bloque con la información que va a enviar el dispositivo al *topic* definido previamente.

```
//TOPIC STRUCTURE
  now = time(nullptr);
  uint16_t lux;
```

```
  state_reported["deviceName"] = deviceName;
  state_reported["time"] = now;


  lux = Luxometro.readLightLevel();//Realizamos una lectura del sensor
  Serial.print("Luz(iluminancia):  ");
  Serial.print(lux);
  Serial.println(" lx");
  state_reported["light"] = lux;
//TOPIC STRUCTURE
```

En la función **void setup()** se incluye un bloque de inicialización del sensor.

```
// BH1750 SETUP

  Serial.println("BH1750 test!");
// Si no se especifica el mode , por defecto es BH1750_CONTINUOUS_HIGH_RES_MODE
  Wire.begin();
  Luxometro.begin(); //inicializamos el sensor

// BH1750 SETUP
```

De modo similar a los anteriores casos se define el nombre de la cosa, así como el punto de enlace con AWS IoT en el fichero **secrets.h**.

```
#define THINGNAME "IoTESP8266Light"
//punto de enlace de API REST
const char MQTT_HOST[] = "aaaaaaaaaaaaa-ats.iot.eu-west-1.amazonaws.com";
```

Con esta configuración y el sensor BH1750 esta es la información que se recibirá de manera periódica.

```
21:49:48.530 -> Luz(iluminancia):  14 lx
21:49:48.530 -> Sending  [device/light]:
{"state":{"reported":{"deviceName":"BH1750","time":1591213787,"light":14}}}
21:49:48.597 -> Received [device/light]:
{"state":{"reported":{"deviceName":"BH1750","time":1591213787,"light":14}}}
```

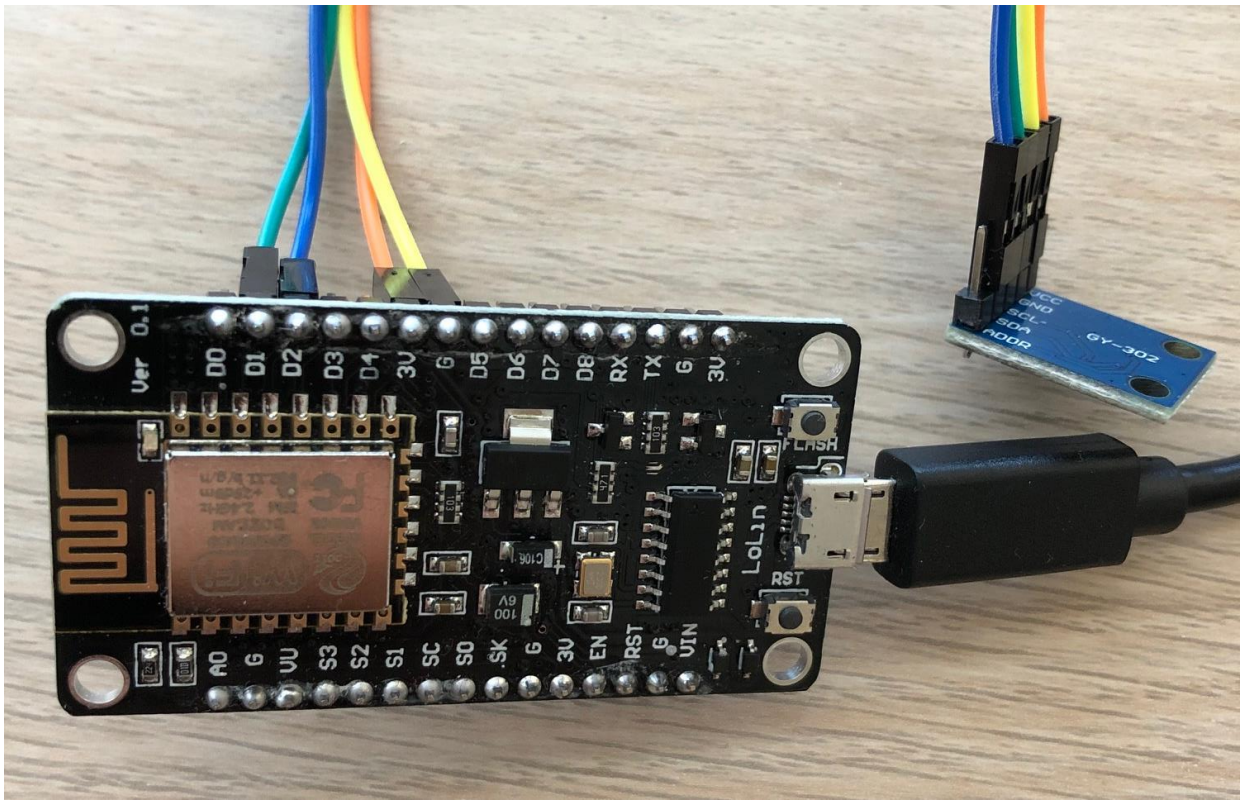


Figura 4-6.Dispositivo ESP8266 con sensor BH1750.

El dispositivo ESP8266 está conectado al sensor BH1750 mediante los siguientes pines.

Dispositivo ESP8266	**Sensor BH1750**
VCC	VCC
GND	GND
D2	SDA
D1 – GPIO 5 utilizado como SCL	SCL

Intervalo de publicación / suscripción

Para los tres tipos de uso de ESP8266 se puede definir el intervalo de tiempo entre publicación de *topics* o de petición de *topic* suscrito. Para ello es necesario modificar la siguientes líneas de la función **void loop()** dentro del fichero **.ino** de cada tipo.

```
    client.loop();
    // 60000 Millis is 60 seconds
    // 3600000 Millis is 60 minutes
    if (millis() - lastMillis > 60000)
//    if (millis() - lastMillis > 3600000)
    {
      lastMillis = millis();
      sendData();
    }
```

4.3.2 Configuración en AWS

Se recomienda la siguiente documentación de referencia [7] (Amazon Web Services, Deploy an End-to-End IoT, 2017)como referencia para el uso del flujo de trabajo de una aplicación en un entorno AWS IoT desde el dispositivo hasta los servicios en la nube.

Comunicación con DynamoDB

Esta información es enviada por una parte a Dynamo DB a través de varias reglas.

En el caso de la medida de temperatura y humedad se crea una tabla IoTESP8266Temperature que va a tener como clave primaria el nombre del dispositivo y como clave de ordenación el momento de recogida de datos. En una columna adicional se almacena el mensaje completo.

La información del *topic* que se quiere enviar a DynamoDB se obtiene mediante la consulta SQL.

```
SELECT state.reported.time, state.reported.temperature, state.reported.humidity FROM
'device/temperature'
```

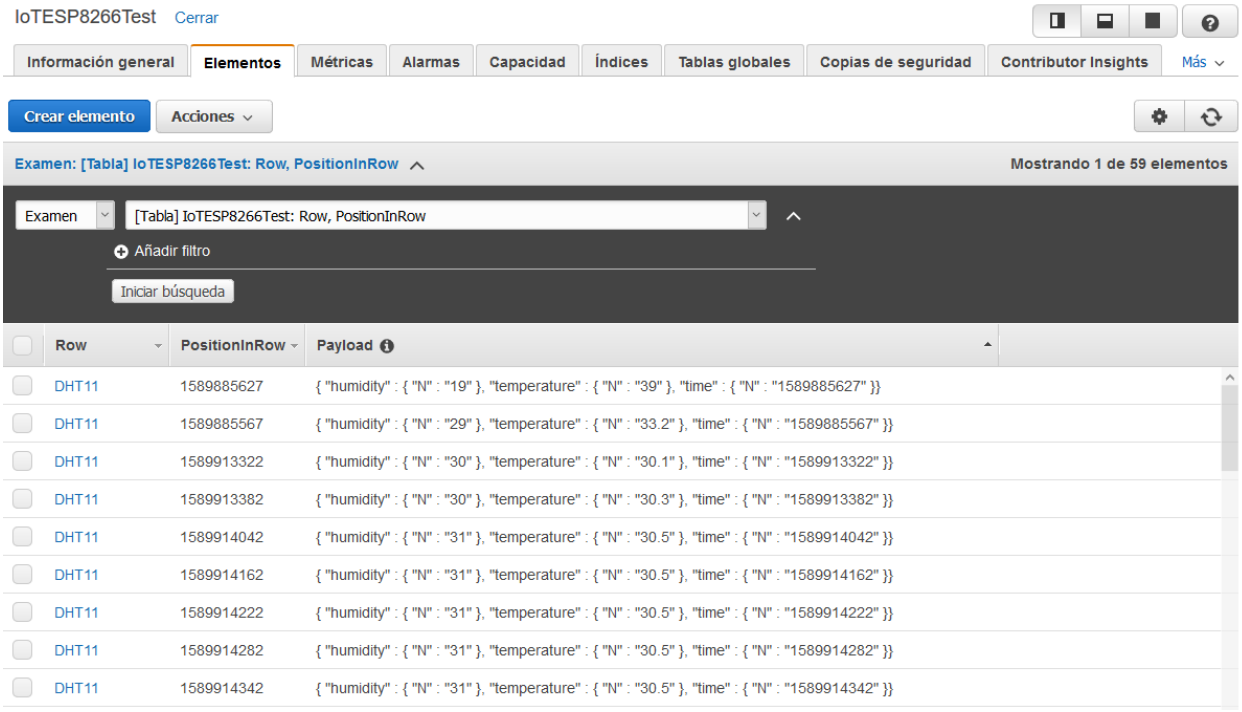


Figura 4-7.Información recibida en DynamoDB del topic device/temperature.

Para la tabla que recoge la información periódica de la luminosidad a lo largo del día denominada ESP8266fromLight se crea una regla que recoge la información del *topic* ***device/light*** e introduce dos claves. La clave primaria con la hora de recogida de la muestra y la de ordenación con el valor en lux de la muestra recogida. El mensaje completo se almacena en la columna *payload*.

Para obtener la información del topic se ejecuta la siguiente consulta SQL.

```
SELECT state.reported.deviceName, state.reported.time, state.reported.light FROM
'device/light'
```

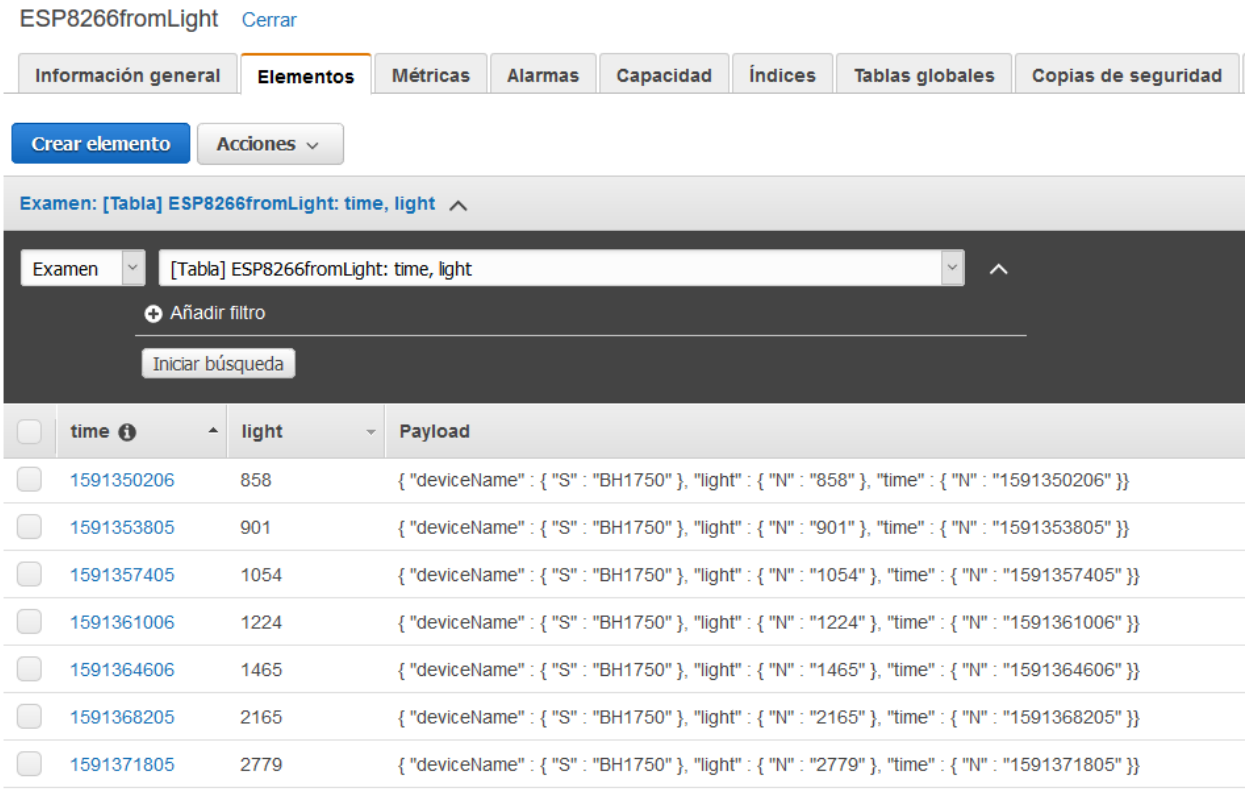


Figura 4-8.Información recibida en DynamoDB del topic device/light.

Creación de función Lambda

Las funciones Lambda se pueden crear través de la consola AWS. Si se utiliza la opción blueprint se puede partir de un esqueleto de función. En el ejemplo se ha seleccionado hello-world-python.

Además de elegir un nombre para la función se puede asignar un rol de ejecución como puede ser la publicación de una notificación SNS: Amazon SNS *publish policy*.

Cada vez que se publica información en el **topic *device/temperature*** se ejecuta la función Lambda. Se procesa para realizar un cálculo aproximado de la sensación térmica y se actúa enviando un correo electrónico en caso de que el valor de sensación térmica supere un umbral.

El procesado de la información recibida relativa a la temperatura y humedad del sensor conectado a través del dispositivo ESP8266 se realiza a través de una función Lambda.

El código específico de la función Lambda para la implementación de la solución programado en Python se muestra a continuación.

```
from __future__ import print_function


import json
import boto3


# initialize Topic Start FAN
client = boto3.client('iot-data', region_name='eu-west-1')


print('Loading function')


def lambda_handler(event, context):
# Parse the JSON message
  eventText = json.dumps(event);
# Print the parsed JSON message to the console. You can view this text in the
# Monitoring tab in the AWS Lambda console or in the Amazon CloudWatch Logs console.
  print('Received event: ', eventText);


# print the resp
  print (event);


# extract an element in the response
  print (event['temperature']);
  print (event['humidity']);


# Convertir a float temperature y humidity


  temp = float(event['temperature']);
  print (temp);
  hum = float(event['humidity']);
  print (hum);
```

```
#Calcular la sensacion termica

  if hum <= 10 and hum >= 0:
     sen = temp - 3
  elif hum <= 20:
     sen = temp - 2
  elif hum <= 30:
     sen = temp - 1
  else:
     sen = temp


  print ('Sensacion termica');
  print (sen);


# Si Sensacion termica > 40 enviar sns topic y publicar en topic fan
  if sen >= 35:
# Create an SNS client
    sns = boto3.client('sns')
# Publish a message to the specified topic
# La sensación termica ha superado el umbral de 35°C. SensacionTermica=xxx
    Mensaje = "La sensación termica ha superado el umbral de 35°C. SensacionTermica = " +
str(sen)
    response = sns.publish ( TopicArn = 'arn:aws:sns:eu-west-
1:462400159810:IoTESP8266_SNS_topic', Message = Mensaje )
    print(response)


# Publish Topic Start FAN
##    client = boto3.client('iot-data', region_name='eu-west-1')
# Change topic, qos and payload
    response = client.publish(
            topic='device/fan',
            qos=0,
            payload=json.dumps({"fan":"1"})
        )
    print(response)
  else:
# Change topic, qos and payload
    response = client.publish(
            topic='device/fan',
            qos=0,
            payload=json.dumps({"fan":"0"})
        )
    print(response)
```

Para el cálculo de la sensación térmica se ha utilizado la siguiente aproximación. La sensación térmica se obtiene combinando los datos de temperatura y humedad relativa de la tabla 4-2.

Temperatura en °C	Humedad relativa en %																				
	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
20	16	16	17	17	17	18	18	19	19	19	19	19	20	20	20	21	21	21	21	21	21
21	18	18	18	19	19	19	19	19	20	20	20	20	21	21	21	22	22	22	22	22	23
22	19	19	19	20	20	20	20	20	21	21	21	21	22	22	22	22	23	23	23	23	24
23	20	20	20	20	21	21	22	22	22	23	23	23	23	24	24	24	24	24	24	25	25
24	21	21	22	22	22	22	23	23	23	24	24	24	24	25	25	25	25	26	26	26	26
25	22	23	23	23	24	24	24	24	24	24	25	25	25	26	26	26	27	27	27	28	28
26	24	24	24	24	25	25	25	26	26	26	26	27	27	27	27	28	28	29	29	29	30
27	25	25	25	25	26	26	26	27	27	27	27	28	28	29	29	30	30	31	31	31	33
28	26	26	26	26	27	27	27	28	28	28	29	29	29	30	31	32	32	33	34	34	36
29	26	26	27	27	27	28	29	29	29	29	30	30	31	33	33	34	35	35	37	38	40
30	27	27	28	28	28	28	29	29	30	30	31	32	33	34	35	36	37	39	40	41	45
31	28	28	29	29	29	29	30	31	31	31	33	34	35	36	37	39	40	41	45	45	50
32	29	29	29	29	30	31	31	33	33	34	35	35	37	39	40	42	44	45	51	51	55
33	29	29	30	30	31	33	33	34	34	35	36	38	39	42	43	45	49	49	53	54	55
34	30	30	31	31	32	34	34	35	36	37	38	41	42	44	47	48	50	52	55		
35	31	32	32	32	33	35	35	37	37	40	40	44	45	47	51	52	55				
36	32	33	33	34	35	36	37	39	39	42	43	46	49	50	54	55					
37	32	33	34	35	36	38	38	41	41	44	46	49	51	55							
38	33	34	35	36	37	39	40	43	44	47	49	51	55								
39	34	35	36	37	38	41	41	44	46	50	50	55									
40	35	36	37	39	40	43	43	47	49	53	55										
41	35	36	38	40	41	44	45	49	50	55											
42	36	37	39	41	42	45	47	50	52	55											
43	37	38	40	42	44	47	49	53	55												
44	38	39	41	44	45	49	52	55													
45	38	40	42	45	47	50	54	55													
46	39	41	43	45	49	51	55														
47	40	42	44	47	51	54	55														
48	41	43	45	49	53	55															
49	42	45	47	50	54	55															
50	42	45	48	50	55																

Tabla 4-2.Sensación térmica en base a temperatura y humedad relativa[3].

[3] Master IoT UCM. Redes Protocolos e Interfaces I. Práctica 2. Bluetooth Low Energy.

- Si la humedad relativa es igual o superior a 0% e igual o inferior al 10% el valor de sensación térmica se calcula restando 3 a la temperatura.
- Si la humedad relativa es superior a 10% e igual o inferior al 20% el valor de sensación térmica se calcula restando 2 a la temperatura.
- Si la humedad relativa es superior a 20% e igual o inferior a 30% el valor de sensación térmica se calcula restando 1 a la temperatura.
- Si la humedad relativa superior a 30% el valor de sensación térmica es igual al de la temperatura.

En caso de superar el umbral AWS IoT envía un mensaje de correo electrónico al usuario mediante el servicio SNS. El correo recibido es el mostrado en la siguiente figura.

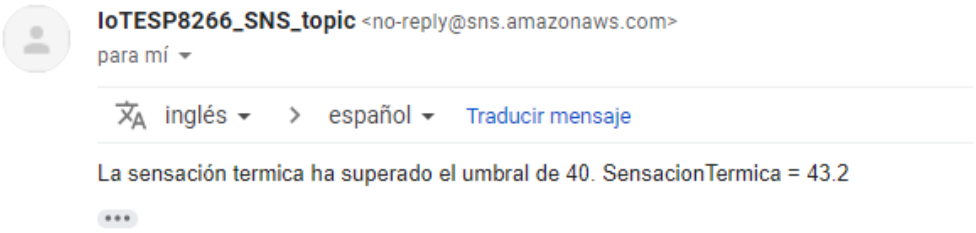


Figura 4-9.Correo recibido a través de SNS.

Con el fin de activar de manera remota el ventilador en caso de que la sensación térmica supere el umbral de 35ºC la función Lambda publica en el *topic device/fan* el valor "0" si debe estar apagado o el valor "1" si debe encenderse según la lógica comentada previamente.

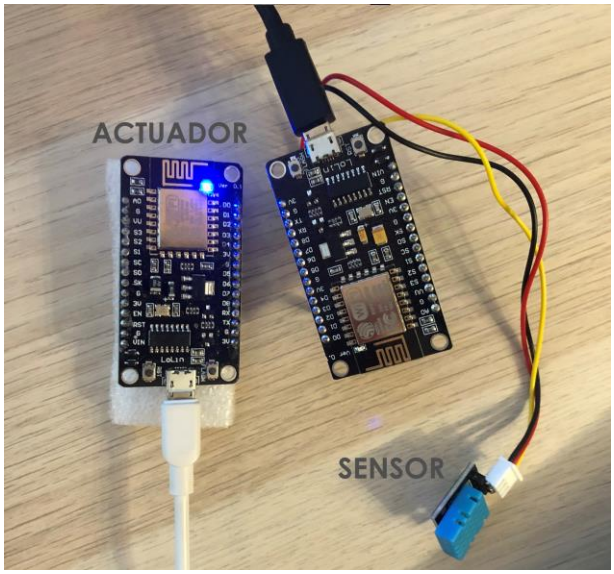


Figura 4-10.ESP8266 como actuador encendido

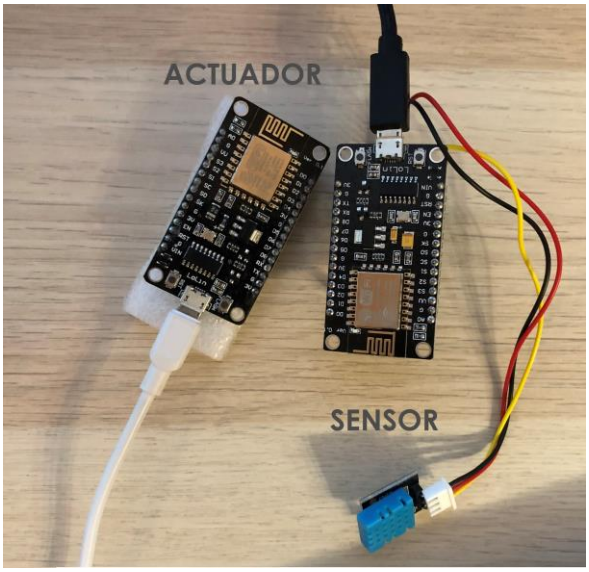


Figura 4-11.ESP8266 como actuador apagado.

En la Figura 4-10. ESP8266 cuando recibe el mensaje {"fan": "1"}. En este caso el led se enciende indicando que el ventilador debe comenzar a funcionar.

21:27:09.920 -> Received [device/fan]: {"fan": "1"}

En la Figura 4-11. ESP8266 cuando recibe el mensaje {"fan": "0"}. En este caso el led se apaga indicando que el ventilador debe pararse.

21:28:09.117 -> Received [device/fan]: {"fan": "0"}

4.3.3 Métricas en AWS IoT

Se puede definir el concepto de métrica en cloud como medición numérica simple que se captura como par de clave y valor. Pueden ser de tipo incremental o agregado. La evaluación de estas métricas puede ayudar a identificar problemas y responder de manera acorde a ellos.

Estos pueden ser algunos de los tipos de métricas que se pueden evaluar en entornos de la nube.

- Tiempo de respuesta

Tiempo que tarda una carga de trabajo en ejecutarse en el entorno cloud.

Es un parámetro que muestra el rendimiento del sistema en la nube y puede tener impacto en la respuesta y disponibilidad.

- Rendimiento por ancho de banda

Rendimiento de ciertas tareas de un servicio de computación en un tiempo específico. Para sistemas que procesan grandes cantidades de datos como pueden ser video o audio se puede medir con la ratio de datos por ejemplo Mbps. En servidores web este cálculo se realiza por el número de usuarios soportados o actividad.

Esta métrica adquiere especial importancia en aplicaciones donde el rendimiento es clave como pueden ser datos de video o científicos, flujos de datos en IoT o información en tiempo real en sistemas de *big data*.

- Capacidad

Se considera la capacidad como la cantidad de carga de trabajo comparada con la infraestructura disponible.

La capacidad promedio se puede calcular estudiando la utilización durante diferentes tiempos de cargas de trabajo con demandas variables.

Sirve para balancear entre oferta y demanda. Cualquier petición en la nube necesita asegurarse de que tiene capacidad para entregar el servicio requerido.

- Escalabilidad

Esta métrica evalúa la capacidad de un sistema para crecer en base a un incremento del uso. Esta escalabilidad se puede basar en el incremento en caso de usuarios simultáneos. Se puede evaluar cuando se llega por ejemplo a un porcentaje máximo de su capacidad.

Que el sistema sea capaz de gestionar gran número de peticiones simultaneas y escalar en caso necesario es una de las ventajas de la elasticidad que proporciona la computación en la nube.

- Latencia

Se puede medir como el intervalo de tiempo que transcurre entre que un dato es transmitido en el origen (en IoT un dispositivo) y su llegada hasta el destino.

Si utilizamos como referencia la transmisión de información en internet un valor de latencia promedio puede ser de alrededor de 36 milisegundos en tráfico local y de unos 73 milisegundos en tráfico transoceánico.

La latencia mide como de buena es la comunicación de dispositivos. Esta medida es incluso menos predecible y más complicada de medir en entornos cloud.

- Tiempo de ida y vuelta (RTT, *round-trip time*)

El tiempo que tarda una petición en ser enviada desde una fuente a un destino y su respuesta en volver a la fuente original. En el caso de uso que se estudiará posteriormente se puede considerar este valor como el tiempo que transcurre entre que un *topic* se publica desde el dispositivo ESP8266 y esta información es devuelta en forma de suscripción al dispositivo.

- Coste por uso

Aquí se puede evaluar el coste de poner en marcha el servicio en entono cloud. Este coste incluye desde los servicios de ingeniería, soporte, atención al cliente, así como los gastos relacionados con la infraestructura física y de requerimientos del sistema.

AWS proporciona herramientas que sirven para la evaluación de diferentes parámetros respecto a la utilización del servicio. Estas métricas y dimensiones son enviadas a CloudWatch cada minuto.

Amazon CloudWatch es un servicio de monitorización y evaluación que se puede utilizar con el propósito de desarrollar, administrar o dar soporte a sistemas en AWS. Ofrece información y datos que son procesados para la monitorización del uso de recursos. Como actuación ante esta información se pueden configurar alarmas de servicio y notificaciones.

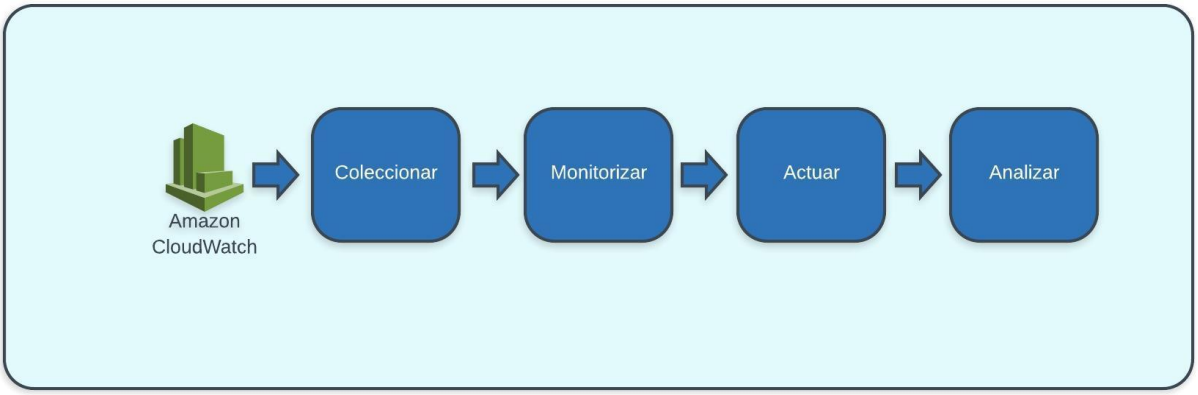


Figura 4-12.Flujo genérico de AWS CloudWatch.

En el caso de uso que se expone en este trabajo se van a evaluar las diferentes métricas relacionadas con los servicios DynamoDB y AWS Lambda.

Las métricas están accesibles desde la consola de AWS en el apartado de CloudWatch.

La tabla 4-3 muestra las diferentes métricas que se pueden comprobar para el caso de uso que se está evaluando.

Métrica	**Descripción**
DynamoDB	
SuccessfulRequestLatency	Información sobre tiempo transcurrido y las peticiones procesadas.
ConsumedReadCapacityUnits	Número de unidades de capacidad de lectura consumidas en un periodo de tiempo específico.
ConsumedWriteCapacityUnits	Número de unidades de capacidad de escritura consumidas en un periodo de tiempo específico.
ReturnedItemCount	Número de elementos devueltos en una operación de consulta.
Acciones de reglas	
Failure	Número de llamadas a una acción que finalizaron en error.
Success	Número de llamadas a una acción que finalizaron correctamente.

Tabla 4-3.Métricas en AWS IoT para el caso de uso de sensor de temperatura.

4.3.3.1 Métricas de AWS DynamoDB

Desde CloudWatch se puede tener una visión general de la infomación relacionada con DynamoDB: tablas creadas, diferentes claves de partición.

Figura 4-13.Alarmas y métricas de DynamoDB en CloudWatch.

Esta visión también facilita la capacidad de lectura / escritura con las que fueron creadas las tablas, sus tamaños y el número de elementos que la componen.

Tablas de DynamoDB

Nombre	Estado	Clave de parti...	Clave de parti...	Capacidad de ...	Capacidad de ...	Tamaño	Item Count
IoTESP8266Te...	Active	deviceName	time	5	5	0 Bytes	0
IoTESP8266Test	Active	Row	PositionInRow	5	5	4.5 kB	59

Figura 4-14.Información sobre tablas en DynamoDB.

AWS/DynamoDB – SuccessfulRequestLatency

La métrica muestra las peticiones procesadas a DynamoBD para un periodo de tiempo específico. Va a proporcionar dos informaciones diferentes:

- Tiempo transcurrido para una petición procesada medido en milisegundos.

- Número de peticiones procesadas.

Los valores que muestra hacen referencia a las operaciones dentro de la base de datos DynamoDB sin tener en cuenta la latencia de red o la actividad dentro del dispositivo.

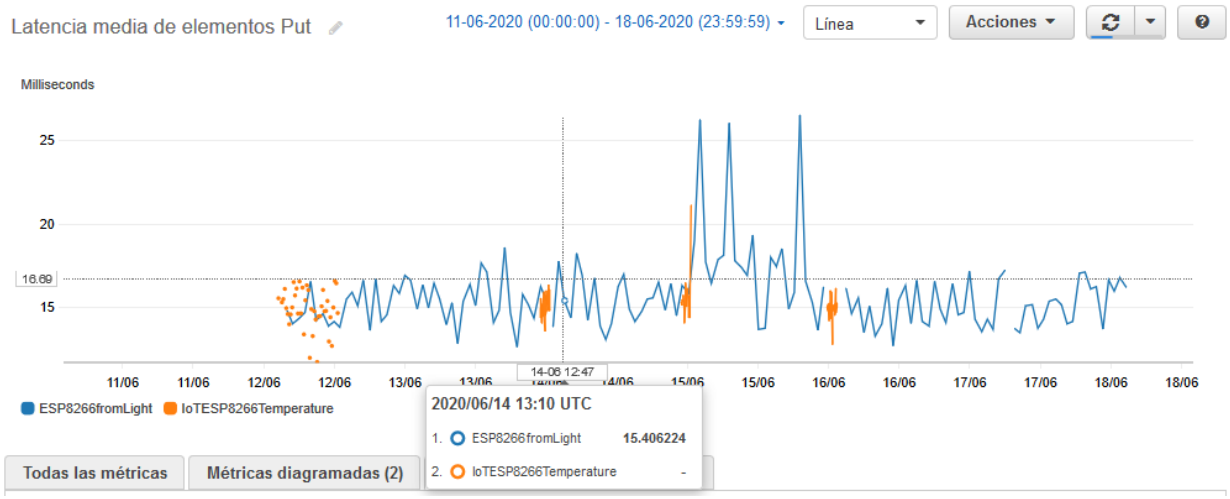


Figura 4-15.Latencia media de elementos PUT.

De la misma métrica se pueden obtener los valores de latencia media de elementos GET.

AWS/DynamoDB – ConsumedReadCapacityUnits

Con esta métrica se obtiene el número de unidades de capacidad de lectura consumidas en un intervalo de tiempo específico. Se puede controlar el rendimiento que ha sido necesario provisionar.

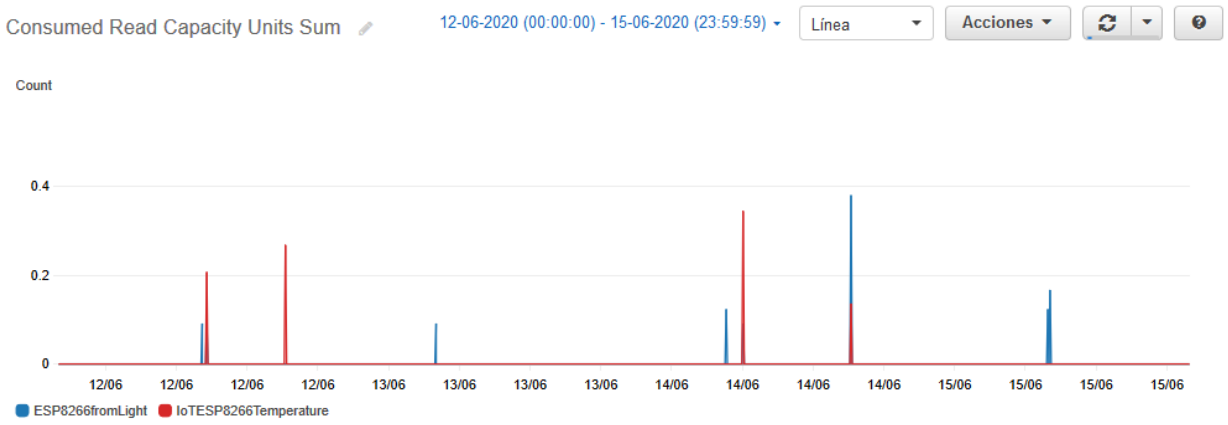


Figura 4-16.Unidades de capacidad de lectura consumidas.

De manera similar se puede obtener el número de unidades de capacidad de escritura consumidas a través de la métrica *ConsumedWriteCapacityUnits*.

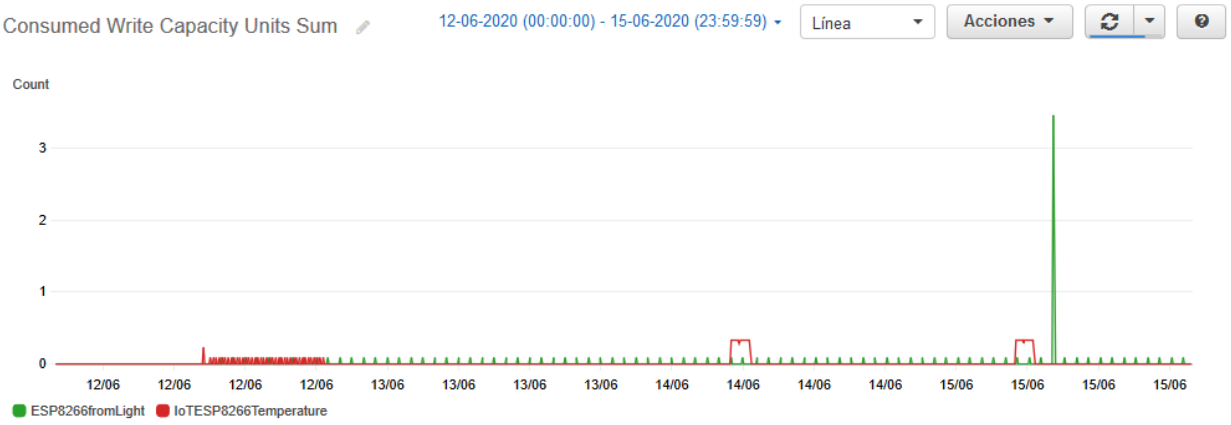


Figura 4-17.Unidades de capacidad de escritura consumidas.

AWS/DynamoDB – ReturnedItemCount

Es el número de elementos devueltos en una operación de consulta o escaneo en un intervalo de tiempo determinado. El número de elementos devueltos no tiene por qué coincidir con el número de elementos evaluados. Se puede especificar como filtro un valor inferior al número total de elementos de la tabla.

4.3.3.2 Métricas en AWS IoT Rules

Relacionadas con las reglas aplicadas en las acciones de IoT se pueden visualizar métricas respecto al procesada correcto o fallo de aplicación de las reglas configuradas.

- Success: Informa del número de llamadas correctas en la ejecución de una acción de regla. La dimensión RuleName contiene el nombre de la regla que especifica la acción. ActionType contiene el tipo de acción que la invocó.

- Failure: Informa del número de llamadas erróneas en la ejecución de una acción de regla. Las dimensiones RuleName y ActionType contienen el nombre de la regla y el tipo de acción que la invocó respectivamente.

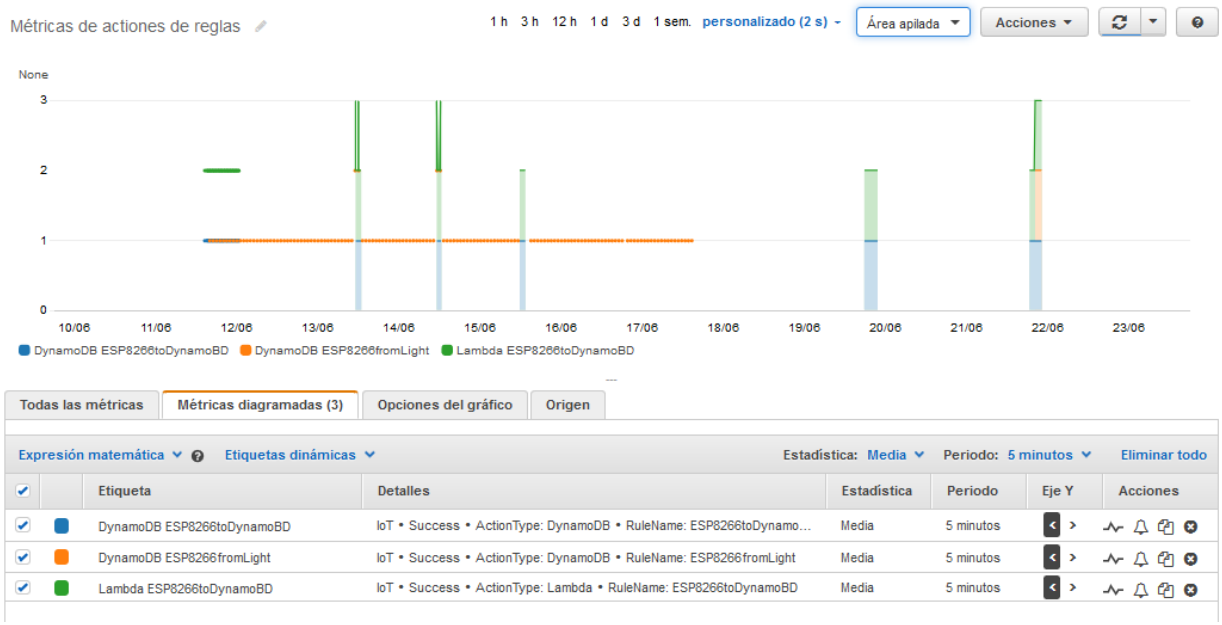


Figura 4-18.Métricas de acciones de reglas.

4.3.3.3 Métricas evaluando el flujo de comunicación entre el dispositivo ESP8266 y cloud.

En la siguiente evaluación se va a comprobar el comportamiento del sistema de punto a punto según se muestra en las figuras 4-1 y 4-2 que describen la arquitectura del

caso de uso "invernadero inteligente" utilizando AWS IoT. Las medidas se van a realizar con el dispositivo ESP8266 conectado al puerto USB de un portátil con el fin de poder obtener en tiempo real a través de puerto COM los valores de la medida y la respuesta a la suscripción.

Se van a utilizar como referencia dos valores de métricas comentados previamente.

- Latencia o tiempo que tarda en publicar el dispositivo un *topic*.
- Tiempo de ida y vuelta (RTT) que tarda un *topic* en ser publicado y posteriormente recibido por el dispositivo como suscripción.

También se va a realizar un estudio del flujo punto a punto (end to end) mediante la medida de una muestra en dos situaciones diferentes. Cuando la sensación térmica no supera el umbral, no se produce un envío de correo electrónico de manera automática a través del servicio SNS. En caso de superar el umbral definido si se produce este envío de correo.

Estudio del flujo punto a punto entre sensor de temperatura y usuario final.

Para evaluar el flujo de comunicación entre el dispositivo con sensor térmico y el servicio IoT de AWS se van a tomar dos valores de medición uno sin superar el umbral y otro cuando el valor supera los 40ºC. Se van a capturar los metadatos de información del correo electrónico para comprobar la fecha y hora de recepción.

La respuesta del sistema cuando el dispositivo está conectado a una red Wi-Fi de alta velocidad es la siguiente.

- **Lectura de temperatura en el dispositivo.**

En el primer caso la sensación térmica no supera el umbral de 40ºC.

```
      13:04:20.213 -> Sending  [device/temperature]:
{"state":{"reported":{"deviceName":"DHT11","time":1590145459,"temperatu
re":26.9,"humidity":41}}}
```

En el segundo caso la sensación térmica tiene un valor de 42,7ºC.

```
      13:09:20.230 -> Sending  [device/temperature]:
{"state":{"reported":{"deviceName":"DHT11","time":1590145759,"temperatu
re":44.7,"humidity":16}}}
```

- **Procesado en AWS IoT.**

Para sensación térmica inferior al umbral de 40ºC.

```
      CloudWatch Log para /AWS/Lambda/ESP8266_lambdaHelloWorld
      START RequestId: afd56981-98f9-4d1f-a55e-8e4ce286eb8d Version:
$LATEST
11:04:20
Received event: {"time": 1590145459, "temperature": 26.9, "humidity":
41}
Received event:
{
    "time": 1590145459,
    "temperature": 26.9,
    "humidity": 41
}
```

Para sensación térmica superior al umbral de 40ºC.

```
      START RequestId: 8b577058-2e5c-4d9a-83ef-53cc4e859d82 Version:
$LATEST
11:09:20
Received event: {"time": 1590145759, "temperature": 44.7, "humidity":
16}
Received event:
{
    "time": 1590145759,
    "temperature": 44.7,
    "humidity": 16
}
```

- **Recepción del topic en el dispositivo desde AWS:**

Para sensación térmica inferior al umbral de 40ºC.

13:04:20.260 -> Received [device/temperature]:
{"state":{"reported":{"deviceName":"DHT11","time":1590145459,"temperatu
re":26.9,"humidity":41}}}

Para sensación térmica superior al umbral de 40ºC.

13:09:20.379 -> Received [device/temperature]:
{"state":{"reported":{"deviceName":"DHT11","time":1590145759,"temperatu
re":44.7,"humidity":16}}}

- Respuesta recibida por correo. Tiempo de metadatos correo.

Para sensación térmica inferior al umbral de 40ºC.

N/A

Para sensación térmica superior al umbral de 40ºC.

Creado a las: 22 de mayo de 2020, 13:09 (entregado en 1 segundo)
De: IoTESP8266_SNS_topic no-reply@sns.amazonaws.com
Para: usuario@ucm.es
Asunto: AWS Notification Message
Delivered-To: usuario@ucm.es
Received: by 2002:ac8:1699:0:0:0:0:0 with SMTP id r25csp199927qtj;
Fri, 22 May 2020 04:09:21 -0700 (PDT)
La sensación térmica ha superado el umbral de 40. SensacionTermica = 42.7

Se puede observar en la información obtenida en las dos pruebas anteriores que el tiempo que ha tardado un suscriptor en recibir el *topic* desde que este es publicado es de 0.047 segundos para el primer caso y de 0.149 segundos cuando supera el umbral

de 40ºC. En esta segunda situación, el mensaje de correo electrónico es recibido en un tiempo inferior a 2 segundos desde que se ha producido la publicación del *topic*

La tabla 4-4 muestra el tiempo en los diferentes elementos del flujo punto a punto: ESP8266 como publicador y como suscriptor, función Lambda y recepción del correo electrónico. Posteriormente se va a realizar un estudio más detallado de los tiempos de publicación y suscripción obteniendo el RTT para diferentes medidas.

Mensaje	Publicación ESP8266	Suscripción ESP8266	Función Lambda	Correo Electrónico
SensacionTermica=41.2	18:20:21.848	18:20:22.151	2020-06-20T18:20:22.106	Sat, 20 Jun 2020 16:20:22
SensacionTermica=44	18:17:21.847	18:17:22.120	2020-06-20T18:17:22.209	Sat, 20 Jun 2020 16:17:22
SensacionTermica=37.5	18:21:21.877	18:21:22.150	2020-06-20T18:21:22.128	Sat, 20 Jun 2020 16:21:22
SensacionTermica=38.7	18:25:21.872	18:25:22.147	2020-06-20T18:25:22.059	Sat, 20 Jun 2020 16:25:22
SensacionTermica=36.2	18:26:21.876	18:26:21.978	2020-06-20T18:26:22.041	Sat, 20 Jun 2020 16:26:22
SensacionTermica=40.3	18:27:21.898	18:27:22.167	2020-06-20T18:27:22.084	Sat, 20 Jun 2020 16:27:22
SensacionTermica=35.8	18:28:21.883	18:28:22.122	2020-06-20T18:28:22.097	Sat, 20 Jun 2020 16:28:22

Tabla 4-4.Tiempos de publicación, suscripción, registro Lambda y correo electrónico.

La función Lambda que se ejecuta en cada recepción de mensaje genera diferentes registros de log en CloudWatch de AWS.

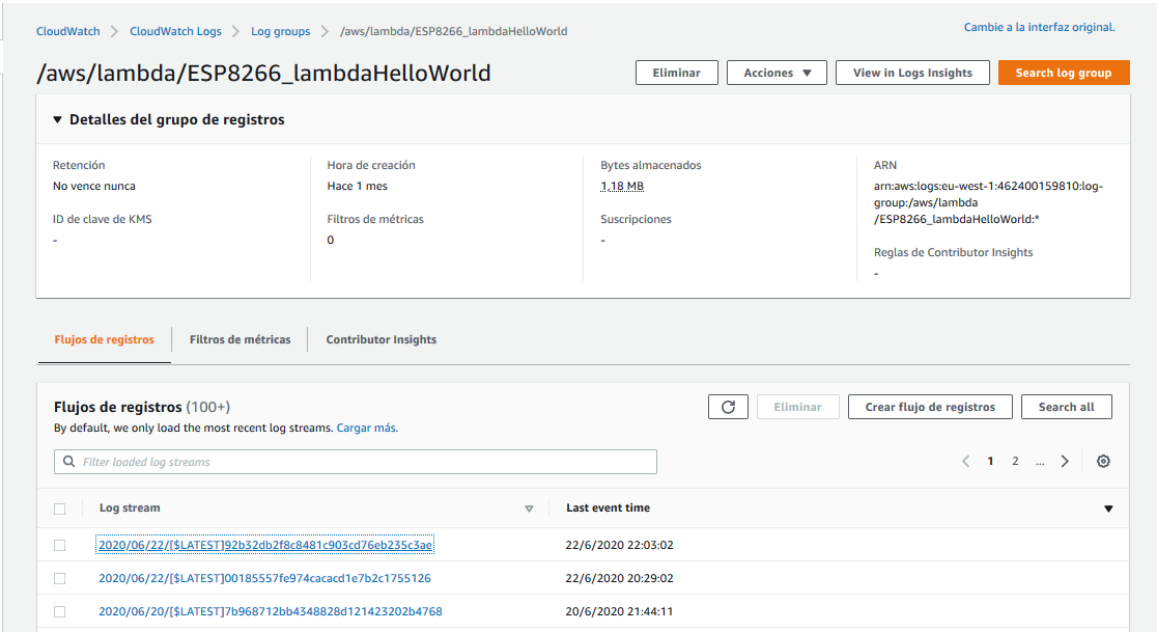


Figura 4-19. Registro de ejecución de la función Lambda en Cloudwatch.

También se puede comprobar la métrica de tiempo de ejecución de la función Lambda para diferentes intervalos como muestra la figura 4-20. En ella se observa que la duración media de ejecución suele ser inferior a 50 milisegundos.

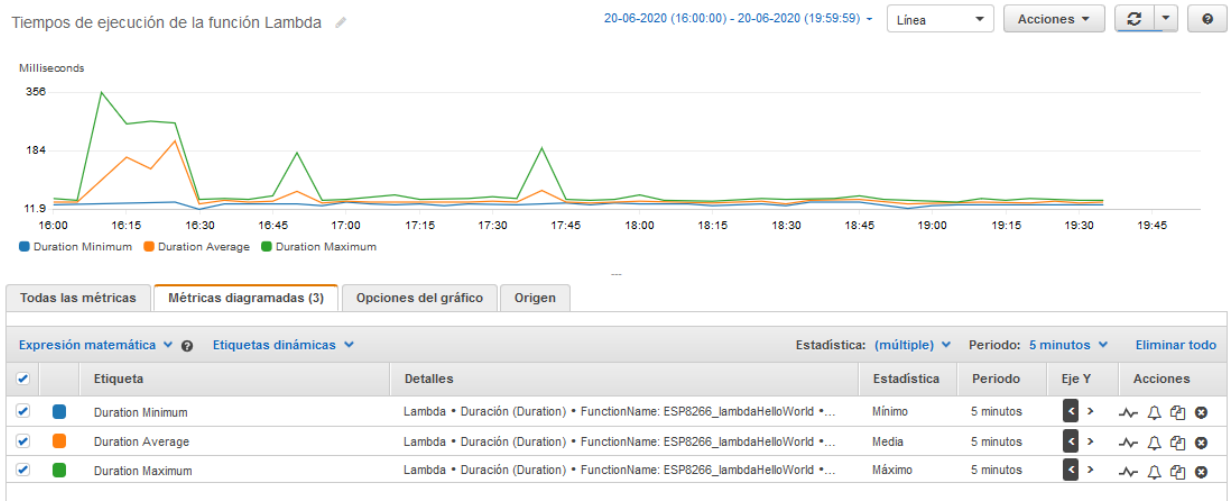


Figura 4-20. Registro de ejecución de la función Lambda en Cloudwatch.

Comparación de Tiempo de ida y vuelta (RTT) en diferentes condiciones de conexión a la nube.

Para este ejercicio se va a comprobar la evolución de tiempos de respuesta a la publicación de un *topic* desde el dispositivo ESP8266. En el primer escenario el dispositivo está conectado a una red de alta velocidad mediante un proveedor de internet con punto de acceso de fibra óptica. Las medidas se han tomado en varios días consecutivos. Para los siguientes casos se hace uso de la red de telefonía móvil 4G. Las medidas se han realizado en dos sitios diferentes.

Las medidas se realizaron con el dispositivo ESP8266 conectado al puerto USB de un portátil para obtener información en local de tiempos de respuesta.

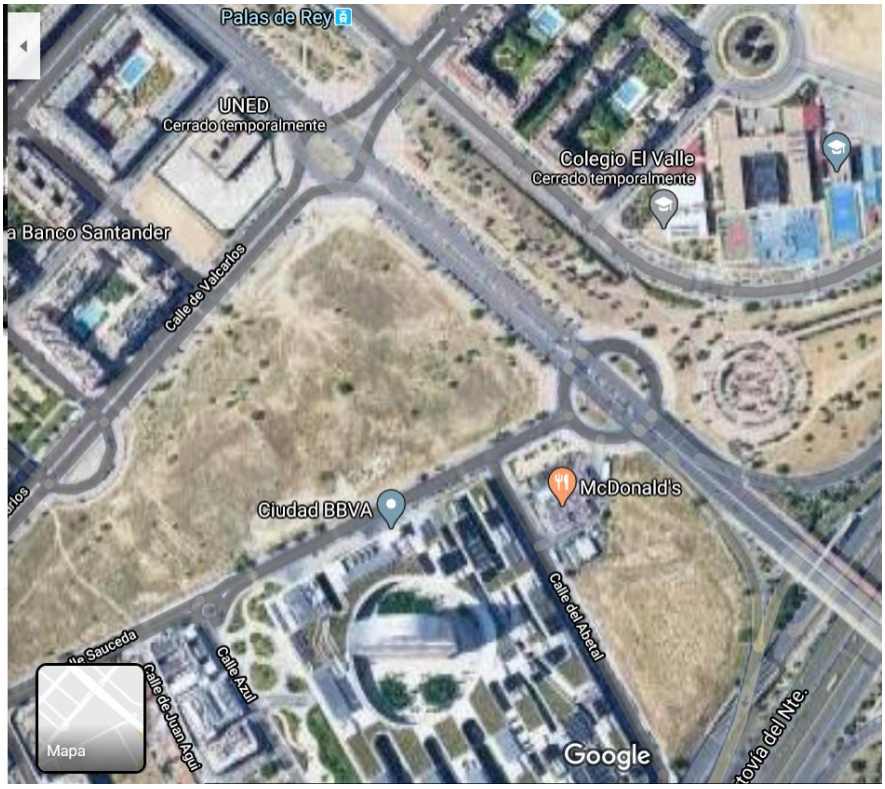


Figura 4-21. Sitio 1: Madrid capital. Zona urbana.

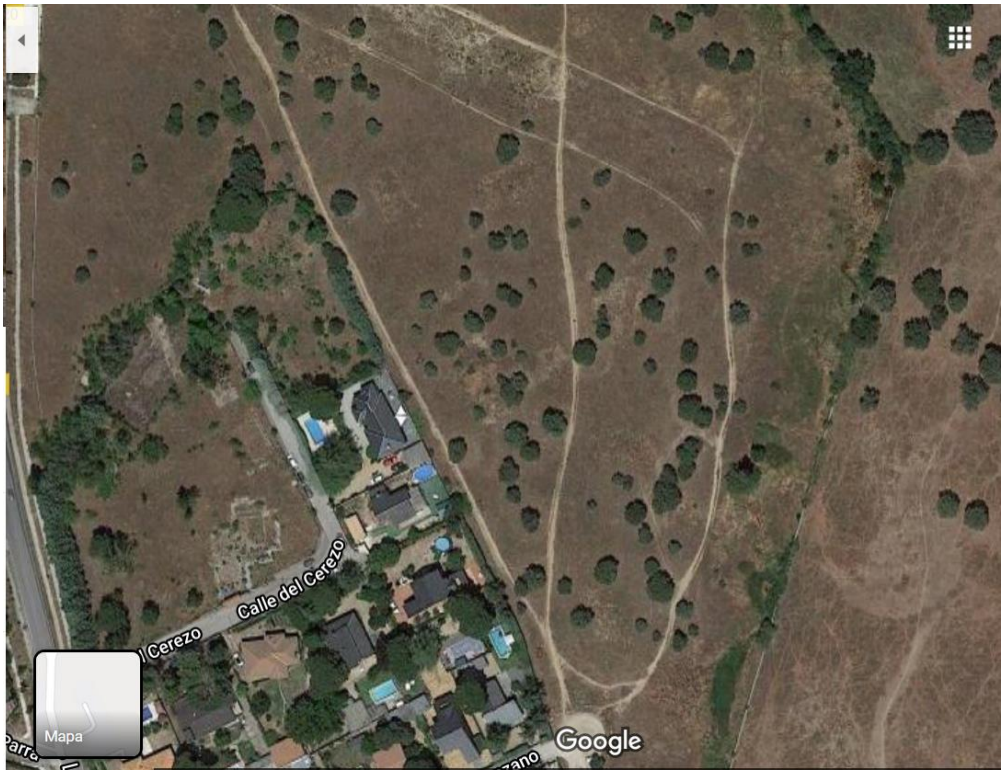


Figura 4-22. Sitio 2: Alpedrete. Zona rural.

La tabla 4-5 incluye un ejemplo de muestras obtenidas en un intervalo de tiempo de 5 minutos.

Tiempo	Dirección	Topic	Información
14:05:28.316	Sending	[device/temperature]:	{deviceName:DHT11,time:1590494727,temperature:29.4,humidity:41}
14:05:28.386	Received	[device/temperature]:	{deviceName:DHT11,time:1590494727,temperature:29.4,humidity:41}
14:06:28.321	Sending	[device/temperature]:	{deviceName:DHT11,time:1590494787,temperature:29.5,humidity:41}
14:06:28.495	Received	[device/temperature]:	deviceName:DHT11,time:1590494787,temperature:29.5,humidity:41}
14:07:28.311	Sending	[device/temperature]:	{deviceName:DHT11,time:1590494847,temperature:29.5,humidity:41}
14:07:28.380	Received	[device/temperature]:	{deviceName:DHT11,time:1590494847,temperature:29.5,humidity:41}
14:08:28.321	Sending	[device/temperature]:	{deviceName:DHT11,time:1590494907,temperature:29.6,humidity:41}
14:08:28.391	Received	[device/temperature]:	{deviceName:DHT11,time:1590494907,temperature:29.6,humidity:41}
14:09:28.327	Sending	[device/temperature]:	{deviceName:DHT11,time:1590494967,temperature:29.6,humidity:41}
14:09:28.430	Received	[device/temperature]:	{deviceName:DHT11,time:1590494967,temperature:29.6,humidity:41}

Tabla 4-5.Tiempos de publicación/suscripción.

La figura 4-23 muestra un intervalo de tiempo de 30 minutos de la medida RTT mediante el acceso a internet a través de fibra óptica para el sitio 1. El comportamiento observado es estable y con valores inferiores a 0,3 segundos salvo momentos puntuales.

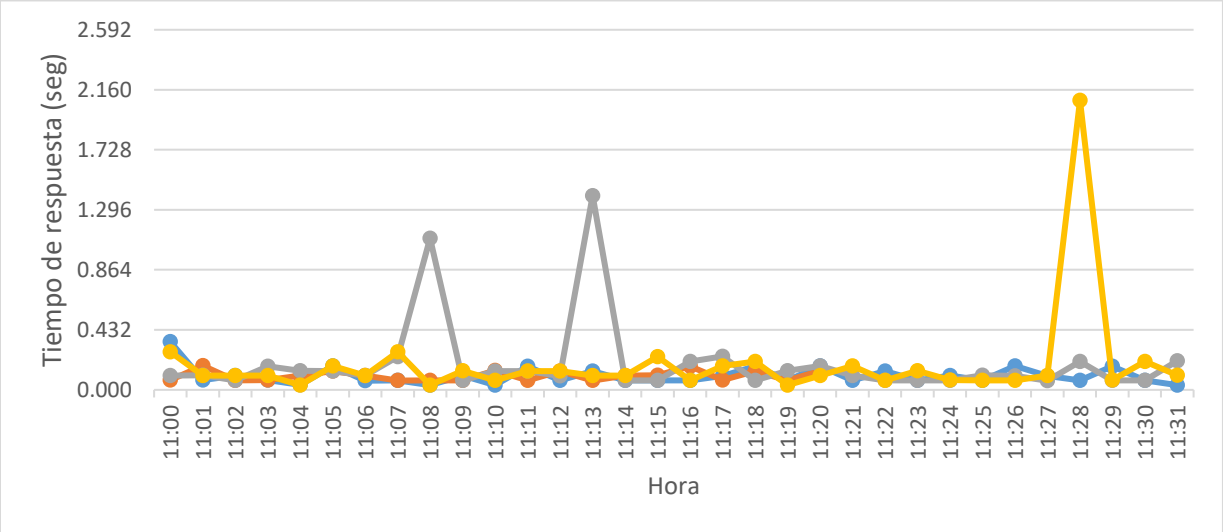


Figura 4-23. RTT del topic. Acceso a cloud mediante fibra. Muestra de diferentes días.

También se ha procedido a una recogida de muestras más amplia realizada durante el mismo día. El acceso a internet también es a través de fibra, aunque esta vez

desde el sitio 2. La figura 4-24 muestra diferentes periodos de muestras obteniendo valores similares a la prueba anterior siempre por debajo de 0,3 segundos.

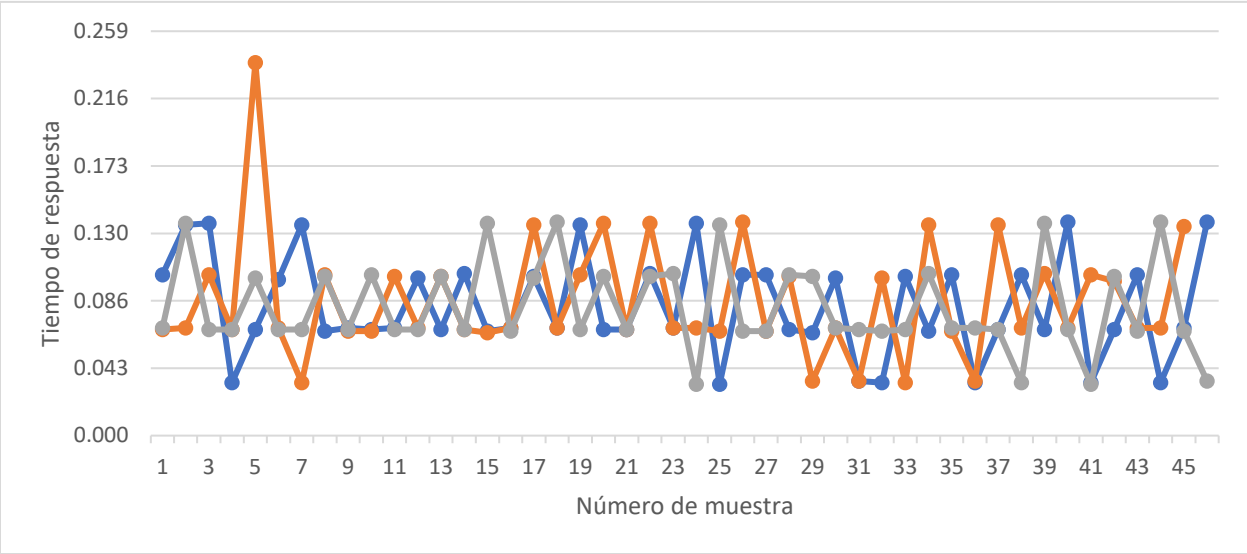


Figura 4-24. RTT del topic. Acceso a cloud mediante fibra. Diferentes muestras durante un día.

La siguiente figura 4-25 muestra las medidas realizadas durante un intervalo de tiempo de 45 minutos aproximadamente. Las medidas han sido tomadas en un entorno urbano. Las muestras se han tomado en días diferentes. La medida RTT es más variable. Se observa un tiempo medio de respuesta algo más elevado que en el caso de fibra.

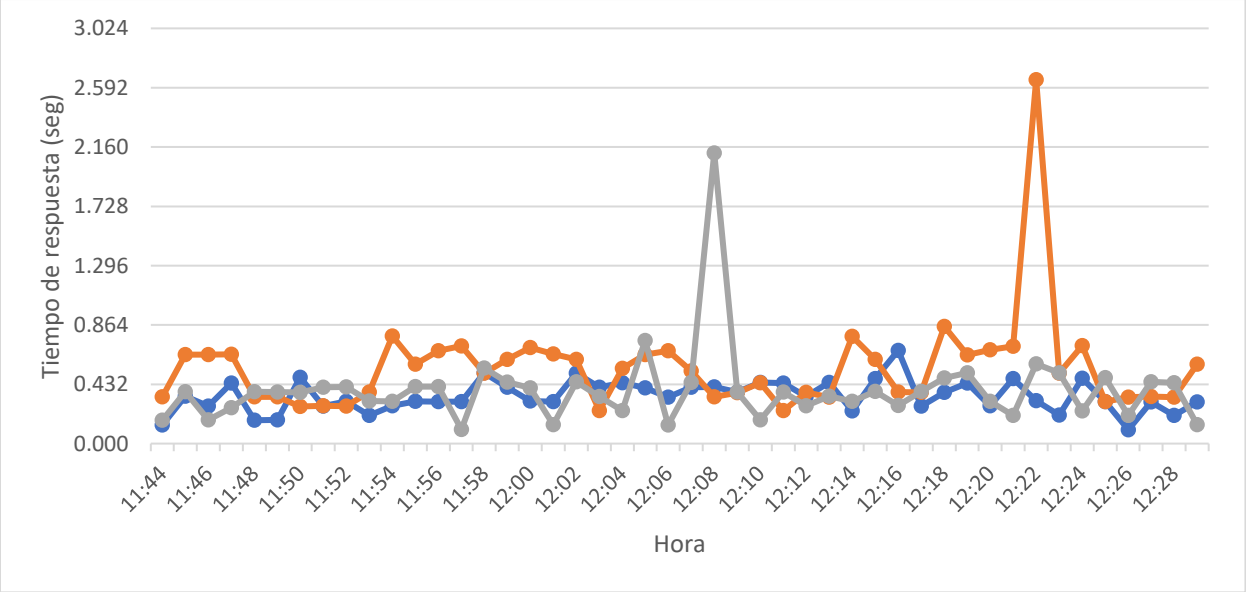


Figura 4-25. RTT del topic. Telefonía móvil – 4G – sitio1.

El segundo sitio es más próximo a un entorno rural, aunque la cobertura móvil es buena. En esta ocasión los valores de las muestras son similares a los obtenidos para un entorno urbano siendo generalmente inferiores a 0,5 segundos según se observa en la figura 4-26.

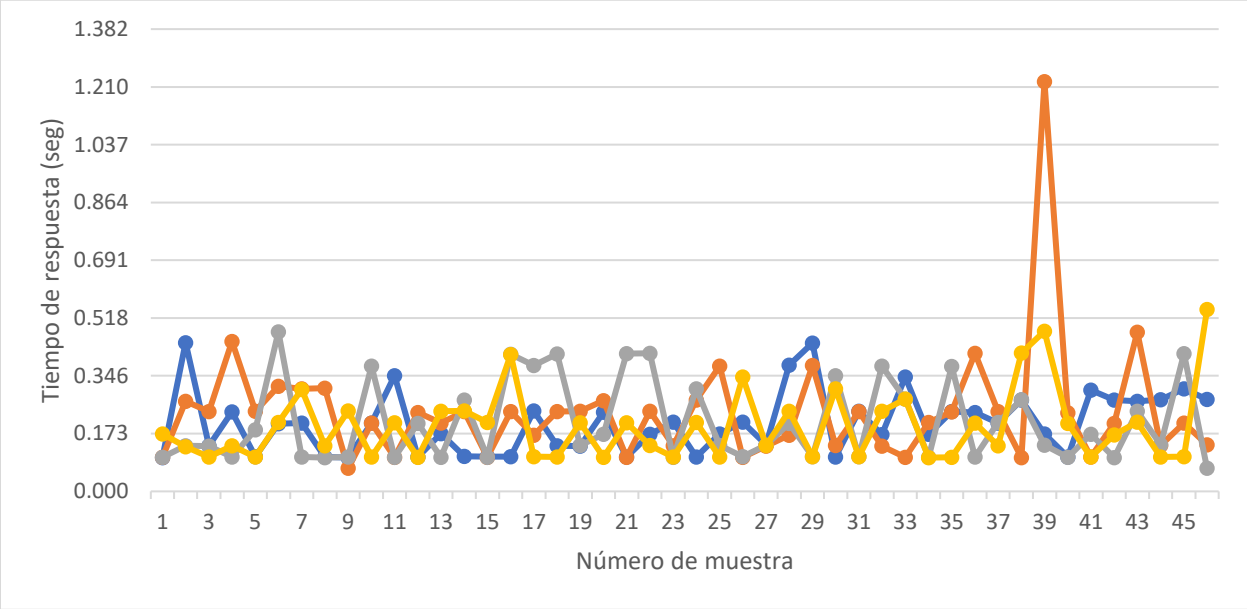


Figura 4-26. RTT del topic. Telefonía movil – 4G – sitio 2.

Aunque se observan diferencias de valores con respecto a la conexión de fibra comparando las latencias y los valores RTT, en el caso de uso de "invernadero inteligente" se puede optar por la solución de Router WiFi 4G dado que es más práctico y sencillo de desplegar que un acceso a internet a través de red de fibra.

Medida de luminosidad mediante el sensor BH1750.

El estudio de la luminosidad del invernadero de cultivo permite comprobar la evolución de la luz del sol a lo largo del día y durante diferentes estaciones del año. En base a estos datos se van a poder tomar diferentes decisiones que afectan al cuidado de las plantas.

- Hora óptima para el riego.
- Apertura de paneles para facilitar el acceso de luz directa.
- Ventilación y oxigenación.

Con el fin de obtener un histórico de la luminosidad se ha creado una tabla en DynamoDB que va a recoger la información publicada en el *topic* ***device/light*** con una frecuencia horaria. Esta tabla aparece en la figura 4-27.

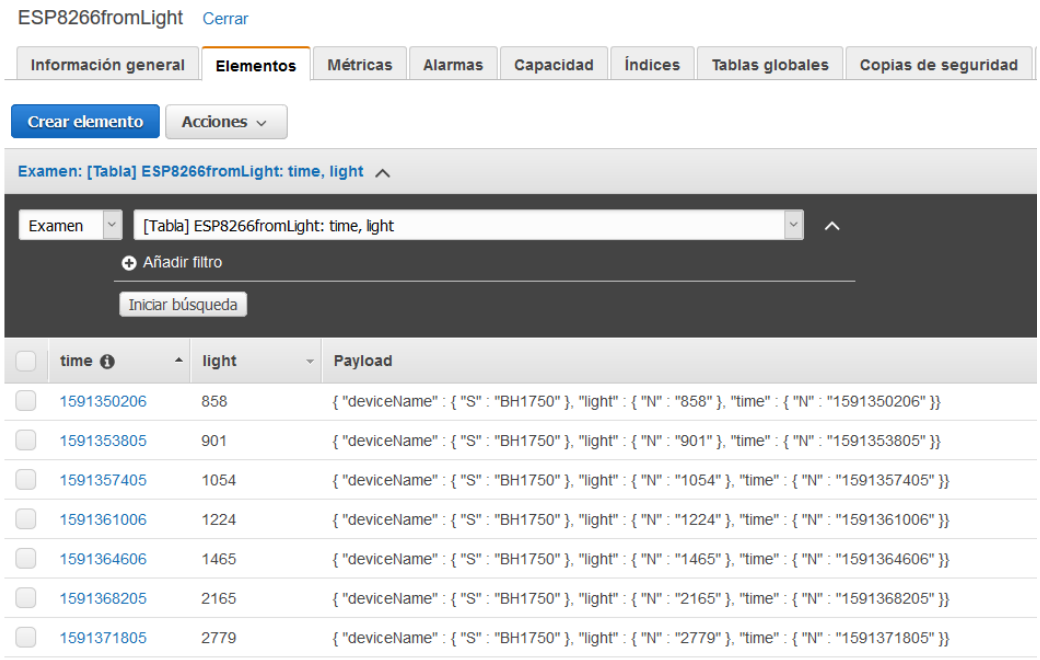


Figura 4-27 Tabla ESP8266fromLight en DynamoDB con la medida de luminosidad.

La figura 4-28 muestra la medida de luminosidad realizada en un intervalo de 60 horas consecutivas en un lugar próximo al sitio 1 descrito anteriormente. Con este ejercicio se consigue por una parte popular la base de datos con información real y por otra parte se pueden obtener conclusiones de la salida y puesta de sol en caso de querer encontrar franjas de tiempo óptimas para el riego.

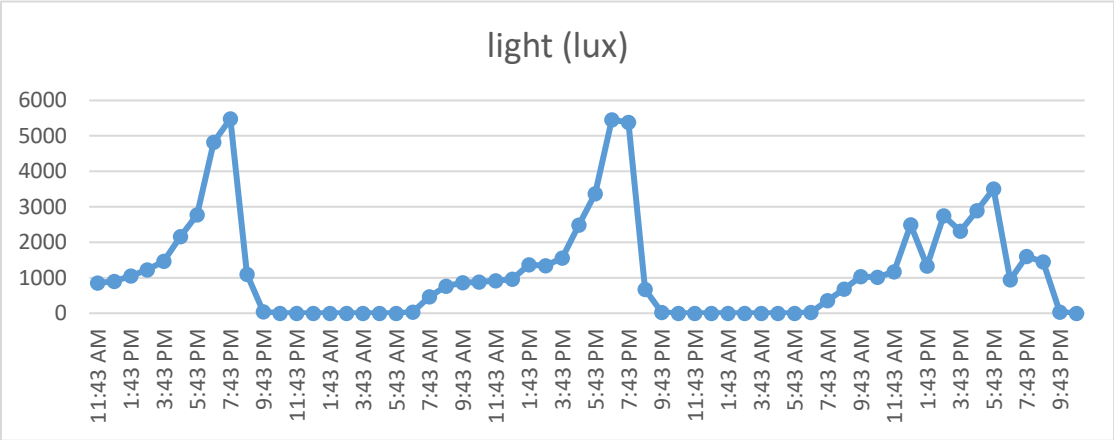


Figura 4-28 Medida de luminosidad (lux) en un intervalo de 48 horas.

Monitorización AWS IoT

La monitorización en AWS para IoT se puede implementar con diferentes aproximaciones dependiendo de la granularidad necesaria. La propia consola de IoT proporciona una monitorización básica.

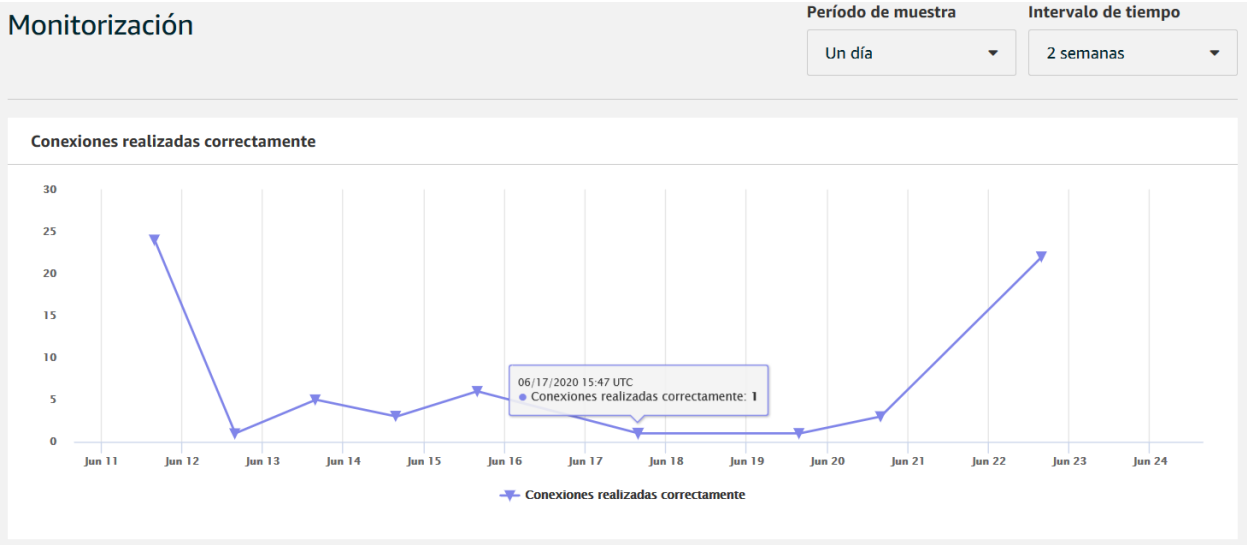


Figura 4-29 Monitorización AWS IoT

Otras posibilidades son AWS *IoT logging* alarmas en *CloudWatch* o logs de uso de las API mediante *CloudTrail*.

IoT logging se puede habilitar para todos los elementos de AWS IoT o sólo para un grupo de cosas específico. A través de *CloudWatch* se puede monitorizar el uso de AWS. La información es almacenada en formato legible durante un histórico de dos semanas. En base a esta información es posible configurar diferentes tipos de alarmas como pueden ser:

- Cuando algún dispositivo o cosa no publica datos diariamente.
- Las actualizaciones de *thing shadow* son rechazadas.
- Si la ejecución de un trabajo programado falla.

Se puede consultar información ampliada en el documento [2] (Amazon Web Services, AWS IoT - Developer Guide, 2019).

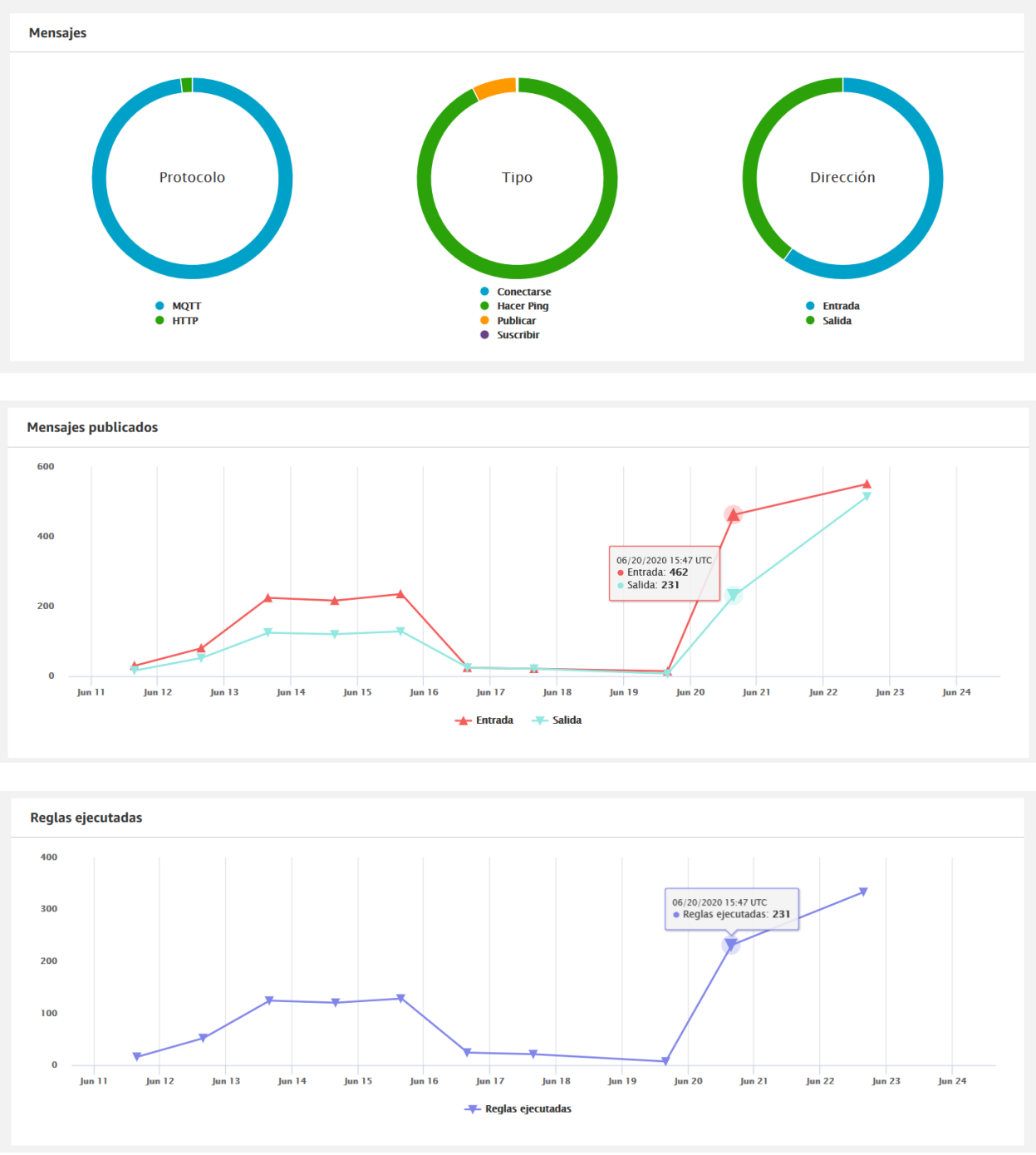


Figura 4-30 Mensajes publicados y reglas ejecutadas.

4.3.4 Costes

4.3.4.1 Evaluación de costes en fase de pruebas y validación.

Durante las pruebas de validación de la solución y toma de datos se ha utilizado un entorno de trabajo con un número reducido de elementos. El diagrama de conexión utilizado aparece descrito en la figura 4-2. Este es el listado de componentes necesarios en la fase de pruebas.

- Dos unidades de sensor de luminosidad BH170. Uno se ha utilizado para las diferentes medidas de luminosidad a lo largo de las horas y días. El otro se ha utilizado para pruebas de conectividad con AWS IoT.

- Una unidad de sensor de temperatura y humedad DHT11. Ha sido utilizado para las pruebas iniciales de conectividad entre el dispositivo y el entorno *cloud* de AWS. También se ha utilizado para las diferentes medidas de tiempo de ida y vuelta o RTT.

- Cuatro unidades de dispositivos NodeMCU ESP8266.

  Un dispositivo se ha conectado al sensor de temperatura DHT11.
  Un dispositivo se ha conectado al sensor de luminosidad BH170 para pruebas de larga duración. El otro dispositivo conectado al sensor BH170 se ha utilizado para pruebas con AWS.
  El ultimo dispositivo se ha utilizado como suscriptor al **topic */device/fan***.

- 4 cables USB para conectar los dispositivos ESP8266 al ordenador portátil o al *power bank*.

- 11 cables tipo Jumper para conectar los dispositivos ESP8266 con los sensores.

- Acceso a red Wi-Fi mediante fibra y también a través de red móvil 4G.

- Entorno de desarrollo Arduino Software (IDE) 1.8.12.

- Cuenta de AWS para el acceso a través de consola a la creación de cosas, tablas de bases de datos, políticas, funciones Lambda, notificaciones y monitorización.

Inicialmente se intentó utilizar una cuenta de pruebas creada mediante la funcionalidad que AWS ofrece a estudiantes[4]. Esta cuenta es bastante limitada en cuanto a la interacción con AWS IoT ya que hace uso de la plataforma vocareum[5]. Por ejemplo, no es posible la creación de llave de acceso o *Access Key* a través del servicio *AWS Educate Starter Account*.

En la tabla 4-6 se pueden observar los costes asociados a la fase de pruebas y validación del sistema.

Elemento	**Comentarios**	**Coste unidad**	**Coste total**
Amazon Web Services			
Debido al uso limitado de servicios y el número de iteraciones con AWS IoT, no se ha repercutido ningún coste desde Amazon Web Services durante la fase de pruebas.			
Hardware			
ESP8266	4 unidades. 2 para los sensores de luminosidad. 1 para el sensor de temperatura/humedad. 1 como simulador de actuador.	5€	20€
Power bank	1 unidad para las medidas de luminosidad.	17€	17€
DHT11	1 unidad.	6€	6€
BH170	2 unidades.	4€	8€
cables tipo Jumper	11 unidades	7€ / 40 ud.	7€
Internet	Acceso a internet mediante las tecnologías de fibra y móvil 4G.	N/A	65€

Tabla 4-6.Desglose de costes por tipo. Fase de pruebas.

[4] https://aws.amazon.com/es/education/awseducate/

[5] https://www.vocareum.com/

4.3.4.2 Evaluación de costes en un escenario real. Invernadero de 4 hectáreas.

Para poder evaluar el coste aproximado de la solución en una situación real se resume las necesidades un campo de cultivo de tipo invernadero tomando como referencia el municipio de Baza, Granada con una superficie aproximada de 4 hectáreas según los datos del documento de la Junta de Andalucía [8] (Consejería de Agricultura, 2017).

Dispositivos, sensores y alimentación.

Estos son los elementos hardware necesarios para llevar a cabo la implementación del caso de uso.

- Sensor de luminosidad BH170: 1
- Sensores de temperatura y humedad DHT11 separados cada 25 metros aproximadamente. 9 dispositivos por 4 hectáreas = 36.
- Actuadores de activación de ventilación 9 x 4 hectáreas= 36.
- Total, ESP8266: 73 unidades.
- Para la alimentación de las unidades ESP8266 se puede utilizar un dispositivo tipo "18650 Battery Shield V8" con voltajes de salida de 3V/1A y 5V/2.2A
- Las pilas recargables de tipo 18650 de Ión-Litio tienen una capacidad de 2,6 Ah. En base a las especificaciones técnicas de ESP8266[6] se toma como valor medio 140mA para el cálculo de consumo. Estos valores ofrecen una duración teórica de 37 horas para cada dispositivo ESP8266. Si los dispositivos transmiten durante una vez cada hora, se estima la duración máxima de la batería de aproximadamente 12 semanas. Se recomienda recargar las baterías con una periodicidad inferior a este valor.

[6] ESP8266EX Datasheet. https://www.espressif.com/en/products/

Tiempo de conexión.

El tiempo de conexión aproximado de cada dispositivo es de 1 minuto por hora. El cálculo para un total de 30 días es el siguiente.

400 conexiones x 1 minuto/hora x 24 horas/día x 30 días = 288000 minutos de conexión.

Coste total de conectividad = 288000 minutos x (0,08 USD / 1000000 minutos) = 0,023 USD.

Mensajes.

Se calcula el coste aproximado de la publicación de mensajes de los dispositivos teniendo en cuenta que el tamaño del mensaje es inferior a 5kb.

400 dispositivos x 4 mensajes/hora x 24 horas/día x 30 días = 1152000 mensajes/mes

Coste total de mensajes = 1152000 mensajes x 1 USD/1000000 = 1,152 USD

Motor de reglas

El número de reglas ejecutadas es aproximadamente el mismo que de mensajes enviados.

Reglas ejecutadas: 1152000 x 0,15 USD / 1000000 = 0,17 USD.

El número de acciones invocadas tendrá un importe similar de 0,17 USD.

Lambda

400 dispositivos x 4 ejecuciones/hora x 24 horas/día x 30 días = 1152000 ejecuciones al mes.

DynamoDB

400 dispositivos x 4 mensajes/hora x 24 horas/día x 30 días x 5kb = 5,76 Gb

La tabla 4-7 lista los elementos descritos previamente necesarios para la implementación del "invernadero inteligente". Los costes referidos al uso de AWS IoT son relativamente bajos debido al número total de mensajes generados mensualmente. Un incremento en la superficie de invernadero a controlar o elevar el número de mensajes por día de los dispositivos elevaría este precio.

Elemento	**Comentarios**	**Coste unidad**	**Coste total**
Amazon Web Services			
DynamoDB	Los primeros 25 GB almacenados cada mes no tienen coste. 0,283 USD por GB-mes para el resto.	N/A	0€
AWS Lambda	Solicitudes: 0,20 USD por un millón de solicitudes. Duración: 0,0000166667 USD por cada GB/segundo. El coste dependerá de la complejidad de las funciones.	N/A	0€
IoT Core	Más de 5 mil millones de mensajes: 0,70 USD (por millón de mensajes)	N/A	1,5 USD/mes
AWS Rules	0,15 USD por millón de reglas activadas o de acciones ejecutadas.	N/A	0,34 USD/mes
Hardware			
ESP8266	73 unidades. 1 para el sensor de luminosidad. 36 para el sensor de temperatura/humedad. 36 como actuador de ventilador.	5€	365€
diymore 18650	Battery Shield V8	10€	730€
Pila 18650	Pila recargable de Ión-Litio, 3.7V, 2.6Ah,	10€	1460€
DHT11	36 unidades.	6€	216€
BH170	1 unidad.	4€	4€
cables tipo Jumper	Pack 120 unidades	7€ / 40 ud.	21€
Internet	Acceso a internet mediante tecnología móvil 4G.	N/A	25€ / mes

Tabla 4-7.Desglose de costes por elemento. Implementación del sistema.

Capítulo 5 - Conclusiones y trabajo futuro

Para la elaboración del trabajo se han utilizado tres elementos principales como son el dispositivo ESP8266 y el entorno de desarrollo Arduino IDE y los servicios en la nube de Amazon Web Services.

En primer lugar, se ha procedido a realizar el estudio del servicio AWS para IoT sus principales funcionalidades y la comunicación con dispositivos. También se ha dedicado una pequeña parte del trabajo a presentar las posibilidades de utilizar los dispositivos ESP8266 mediante una red ESP-MESH. Dentro del apartado práctico de desarrollo de caso se ha realizado una aproximación a una solución de Invernadero inteligente conectando sus dispositivos con los servicios en la nube a través de internet.

La información que se obtiene de los dispositivos y sus diferentes sensores es procesada y almacenada en AWS. Se transforma mediante funciones Lambda a las necesidades específicas del caso de uso. Una de ellas ha consistido en la notificación mediante correo electrónico en caso de sensación térmica elevada y la activación de actuadores. También se procede a almacenar esta información en la nube bien para la elaboración de un histórico de estos datos o bien para su posterior procesado.

La comunicación entre dispositivos y la nube se ha realizado mediante el protocolo MQTT. Para que ésta fuera posible se han generado certificados asociados a las cosas creadas en AWS que son necesarios cargar en los dispositivos. Estos certificados permiten que el dispositivo se autentique de manera segura en AWS IoT y pueda leer y escribir información. El proceso de seguridad en Amazon se ha complementado mediante IAM. La documentación disponible como guías de desarrollo en AWS es muy extensa y está disponible en la página web de Amazon Web Services por lo que se recomienda su consulta en caso de dudas o necesidad de profundizar en alguna funcionalidad.

Los dos elementos principales utilizados en AWS son DynamoDB y funciones Lambda. El primero es el servicio de base de datos que permite almacenar la información de los dispositivos. Para ello se han creado dos tablas. La primera almacena

la información que los dispositivos envían en tiempo real respecto a las medidas de temperatura y humedad. La segunda tabla sirve para almacenar un muestreo de la luminosidad a lo largo de los días.

Obtenidos los resultados de datos, se han estudiado algunas métricas de respuesta del proceso. Por una parte, se han calculado los tiempos de respuesta desde que se publica un *topic* hasta que éste está disponible en el suscriptor. Para esta prueba se han utilizado conexiones a internet tanto de fibra como de datos. En ambos casos los tiempos de respuesta son relativamente similares optándose por la tecnología móvil 4G por la mayor flexibilidad que ofrece respecto a su instalación. También se ha comprobado la información de métricas que el servicio AWS IoT ofrece.

Uno de los principales problemas encontrados durante el desarrollo práctico del caso de uso ha sido la autenticación entre el dispositivo ESP8266 y AWS IoT. Finalmente se ha optado por utilizar certificados generados en AWS y cargados en los dispositivos.

La realización de este proyecto me ha permitido por una parte poner en práctica los conocimientos adquiridos durante la realización del Máster IoT. Al haber elegido herramienta de procesado en la nube AWS he podido incorporar los conocimientos en tecnologías *cloud* obtenidos mediante mi experiencia profesional durante los últimos años.

AWS ofrece una amplia gama de servicios de IoT que no eran objeto de estudio en este trabajo pero que pueden proporcionar estudio y procesado de información obtenida en campo. AWS Kinesis ofrece procesado de datos en tiempo real e incluso la información de los mensajes almacenada en DynamoDB también se puede procesar mediante diferentes funciones Lambda que van a permitir programar código tanto en Python como en Node.js.

Otra línea futura de trabajo es la incorporación de diferentes actuadores y su comunicación con elementos mecánicos como pueden ser sistemas de ventilación o circuito de riego.

Chapter 6 Introduction

The term Internet of Things describes the physical components which communicate through networks and interact with the environment with the help of sensors. This type of interaction allows a change in the way people develop daily activities.

Internet of Things has a wide spectrum of uses. Among others are medical health, energy, farming, homes and Smart cities. An estimation of 30k millions of connected devices are expected by 2020.

6.1 Motivation

These are the four main topics that describe the architecture and utility for Internet of Things.

Low consumption embedded systems. This result in an increment of performance in exchange for low power consumption.

Cloud computing. Using the different platforms available in the cloud such as Microsoft Azure, Google Cloud or Amazon Web Services, the information from these devices can be collected, processed and stored.

Network architecture. Devices can communicate in a data network with other devices, systems or the cloud over the internet.

Data analysis. The large amount of information that can be collected by these devices could be used to obtain a large Flow of data. This information will be used to process results based on data.

The work is focused on two of these four topics. The creation of a network of sensors with ESP8266 devices and the use of the data flows obtained from the devices for storage and cloud process.

ESP8266 devices has been selected because of the reduced size and price but also for the facility for distributed networks creation. With an internet connection via root node and the access to raw data analysis in the cloud the possibilities to add some type of intelligence to the ecosystem to be managed are increased.

6.2 Objectives

- Study of the cloud services provided by Amazon Web Services. Messages reception from devices. Information processing and actions to be taken based on specific events.
- Programing of ESP8266 devices with different functions for sensors and actuators.
- Creation of an intelligent greenhouse ecosystem.
- Costs for the solution implementation.

6.3 Plan of work

First, it is intended to study the solution that AWS provides for IoT. From which are the protocols that can be used to communicate with the cloud to the possible modes of information storage. The official documentation and development guides of Amazon Web Services will be used for this purpose.

A brief description of the ESP-MESH architecture that serve to massively deploy interconnected devices will be done.

Finally, the "**smart greenhouse**" use case will be developed, for which it will be necessary to program the ESP8266 devices with specific characteristics depending on the functionality of each one. Within this chapter, a study of different metrics will also be carried out to evaluate the behavior of the endo to end solution.

The chronology in Figure 6-1 shows the planning that will be used to carry out this work.

	Week number																							
Task	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Technlogies study																								
state of the art																								
ESP and Arduino																								
System design																								
ESP8266 code																								
AWS IoT configuration																								
End to End tests																								
Memory																								
Preliminary version																								
State of the art development																								
Use case development																								
Conclusions																								

Figure 6-1. Working plan.

Chapter 7 Conclusions and future work

Three main elements have been used for this work elaboration: ESP8266 device, development kit Arduino IDE and cloud Amazon Web Services tools.

In first place, a study of AWS services for IoT has been elaborated. This task has included main AWS functionalities and communication within devices and the cloud. This part of the work has included a little research about the of ESP8266 devices and ESP-MESH network technology. The topic for the development of the case of use is the "Smart Greenhouse" connecting devices and sensors with the cloud services using internet for the communication.

Information extracted from devices and sensors is processed and stored in AWS. It is also transformed using the Lambda Function provided by AWS and adapted for the case of use. The features selected for the work described in this document are the notification via e-mail of high thermic sensation and the implementation of actions through actuators. The information received from the devices is also stored in the cloud either for historic tracking of these data or for further processing.

Communication between devices and the cloud is done using the MQTT protocol. To implement this type of communication it is needed to use certificates linked to the things created in AWS. These certificates are loaded into the devices and allows authentication in a secured way in AWS so they can read and write information. The security process is complemented with the use of IAM and roles. The documentation available and development guides are quite extensive, and they are available in the Amazon Web Services web pages. In case a Deep knowledge in specific topics is needed it is recommended to use this information as reference.

The two main elements used in AWS for this work are DynamoDB and Lambda functions. The first one is the data base service that allows store devices information. Two different tables have been created for this purpose. One will store the information received from devices in real time about temperature and humidity. The other table will store the luminosity through different days of the year.

Once this data information has been generated some processing time metrics have been studied. One of these metrics is the time spent by a message since it has been published until it has been received by the subscriber or RTT.

Fiber optic and mobile 4G connections have been analyzed for those tests. In the two cases the response times are similar. 4G technology is the option preferred because of the flexibility in the installation. AWS IoT also offer some metric information for the MQTT communication between the devices and the cloud.

One of the main problems faced during the practical development of the use case have been the authentication between devices ESP8266 and AWS IoT. Finally, the option chosen was to use certificates generated in AWS and loaded into the devices.

While working on this project I have been capable of put in practice the knowledge acquired during the Master of IoT. The use of AWS as the cloud tool also allowed me to sum my professional experience during last days to the exercise.

AWS offers a wide spectrum of IoT services that were not in the scope of study for this work but that can complement the study and process of field information. AWS Kinesis can process data in real time but also the information stored in DynamoDB can be processed offline with different Lambda functions that can be coded either in python or Node.js.

Another future line of work can be the addition of different actuators and its communication with mechanical elements like ventilation or irrigation systems.

BIBLIOGRAFÍA

[1] T. Joshva Devadas, S. Thayammal, A. Ramprakash, Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm, Springer. (2020).

[2] Amazon Web Services, Inc (2019). AWS IoT - Developer Guide.

[3] Amazon Web Services, Inc (2012). Amazon DynamoDB Developer Guide API Version.

[4] Ekaterina Balandina, Sergey Balandin, Yevgeni Koucheryavy, Yevgeni Koucheryavy (2017). IoT Use Cases in Healthcare and Tourism. IEEE 17th Conference on Business Informatics.

[5] espressif « ESP-MESH protocol» [En línea] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/mesh.html [En línea]. [Último acceso: 24 05 2020].

[6] Iván Manuel Laclaustra, Jesús Martín Alonso, Alberto A. del Barrio, Guillermo Botella, Sistema Domótico Distribuido para Controlar el Riego y el Aire Acondicionado en el Hogar. Enseñanza y Aprendizaje de Ingeniería de Computadores. Número 6. (2016)

[7] Amazon Web Services, Inc (2017). Deploy an End-to-End IoT Application.

[8] Consejería de Agricultura, Pesca y Desarrollo Rural. (2017). Cartografía de invernaderos en Almería, Granada y Málaga.

APÉNDICES

Apéndice A - Código de Aplicación para Arduino

El código desarrollado específicamente para la implementación de las diferentes funcionalidades del dispositivo ESP8266 en el presente trabajo, así como el de la función Lambda se encuentra disponible en el siguiente enlace. Puede ser utilizado como licencia de código abierto.

https://drive.google.com/drive/folders/1VVye5a5cxR9rnSSYkn0wnef2BwEyXbJR?usp=sharing

Este código hace uso de diferentes librerías y ejemplos disponibles en internet que se mencionan a continuación.

El código utilizado como referencia para la comunicación MQTT con AWS IoT es el publicado por **debsahu**.

https://github.com/debsahu/ESP-MQTT-AWS-IoT-Core/tree/master/Arduino/PubSubClient

Para la lectura de información de Temperatura y humedad del sensor DHT11 se ha utilizado la librería de **Adafruit**.

https://github.com/adafruit/DHT-sensor-library

Para la lectura de información de Temperatura y humedad del sensor BH1750 se ha utilizado la librería de **claws**.

https://github.com/claws/BH1750

Apéndice B - Configuración del entorno cloud en AWS IoT

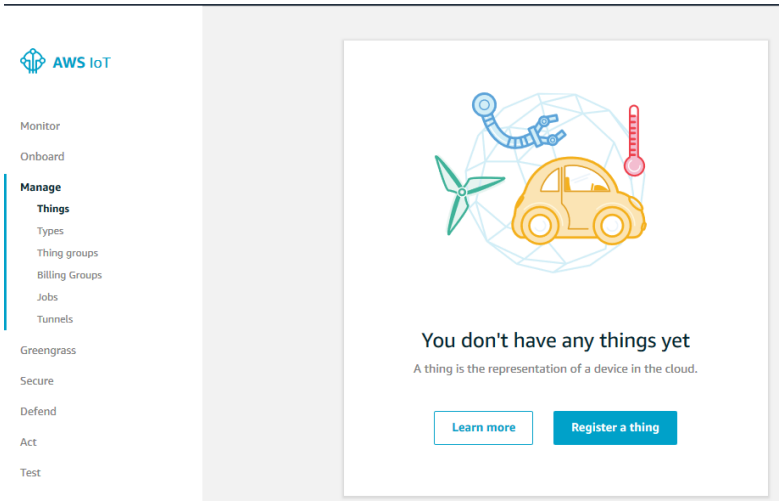
En este apéndice se describen los pasos necesarios para configurar el entorno de pruebas en AWS IoT cloud.

Creación de una cosa en AWS IoT

Los dispositivos conectados a AWS IoT se representan en el registro mediante cosas. Esta cosa representa a un dispositivo o entidad lógica como puede ser un sensor de temperatura o una bombilla.

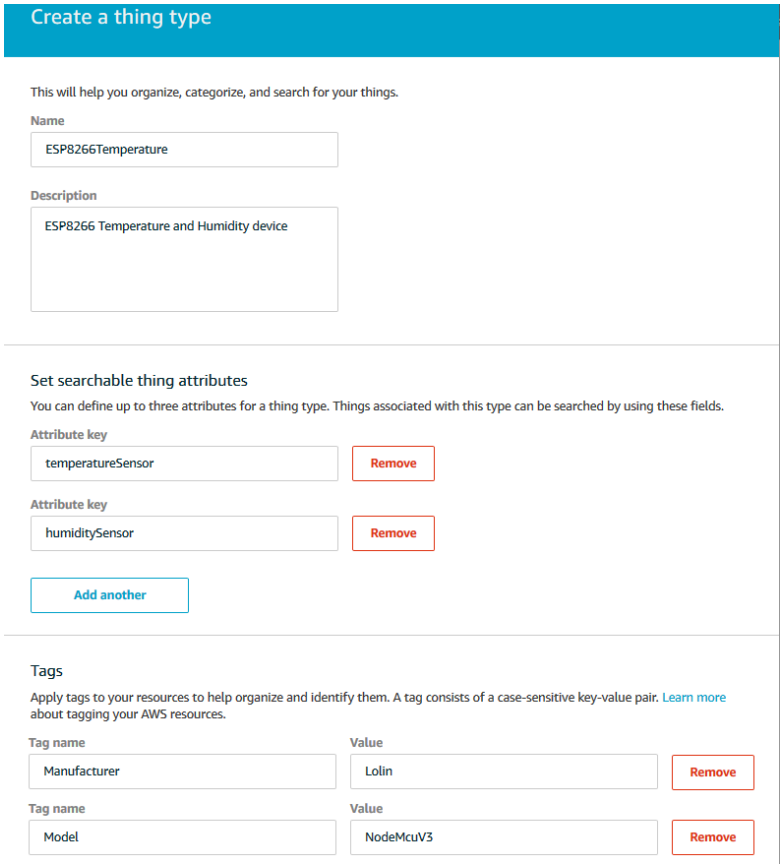
Para crear una cosa hay que ejecutar los siguientes pasos en la consola de AWS IoT.

1) Manage → Things → **Register a thing**.



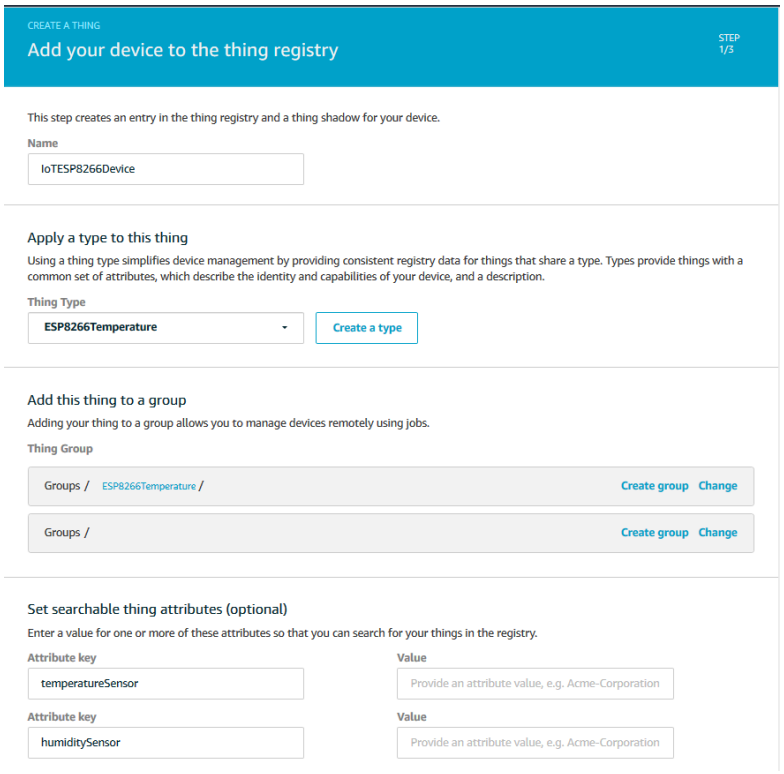
Los tipos de objetos tienen la función de almacenar información descriptiva y de configuración común a todos los objetos.

2) **CreateThingType**.



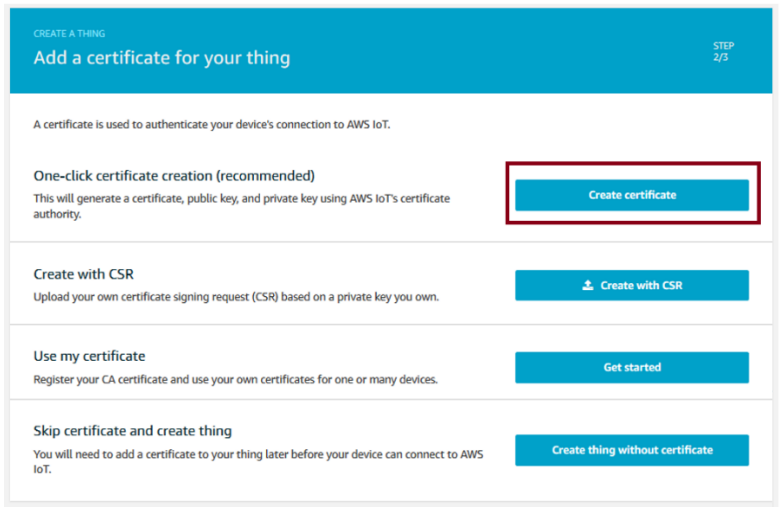
Se puede asociar el tipo a la cosa que se está creando grupos de objetos. De esta manera se administrarán varios objetos a la vez.

3) **CreateThingGroup**. → Add this thing to a group.



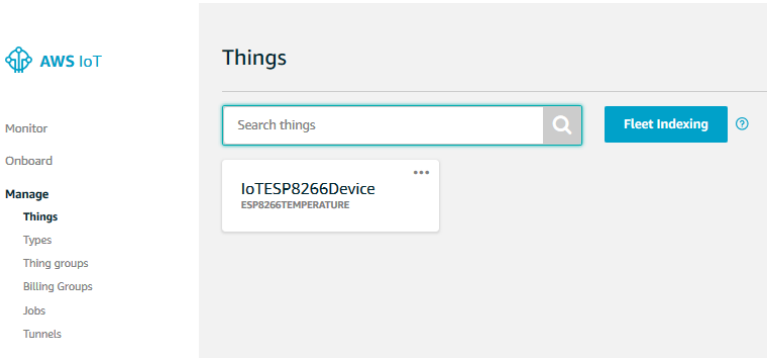
El dispositivo necesita un certificado, clave privada u certificado CA raíz para autenticarse en AWS IoT Core. Se pueden generar los certificados durante el proceso de creación de la cosa y posteriormente incluirlo en el dispositivo.

4) Add certificate to your thing → Create certificate.

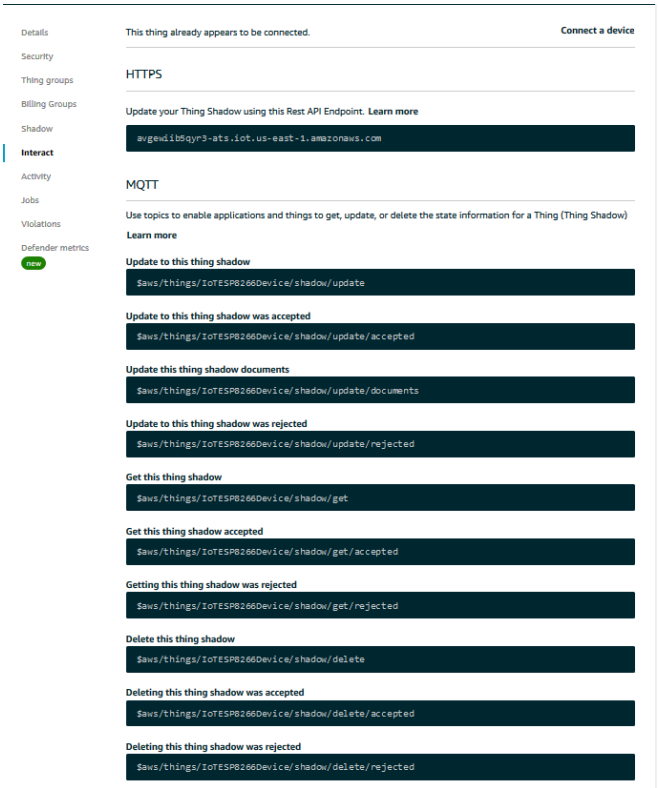


Una vez que la cosa está creada se muede acceder a su configuración de administración en el siguiente menú:

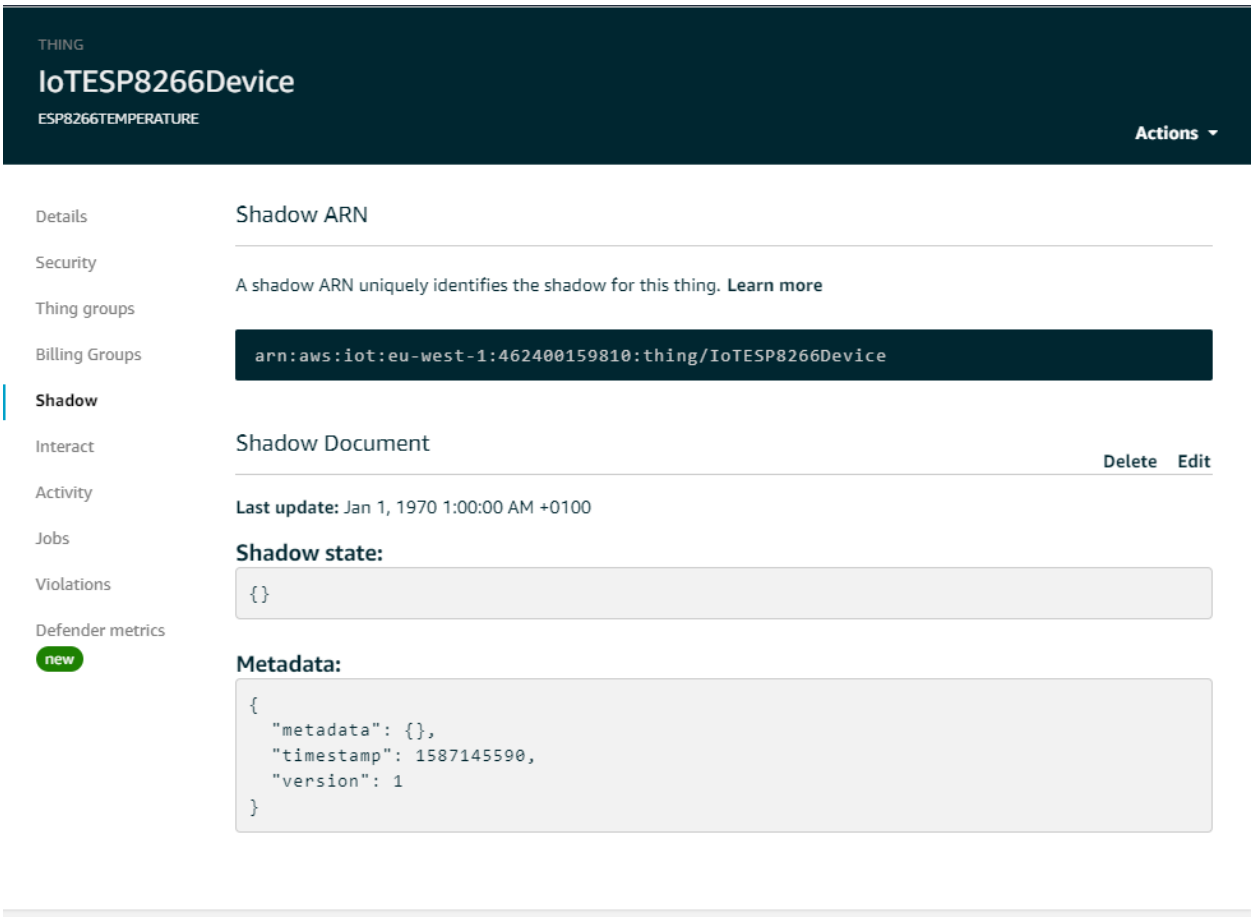
5) Manage → Things.



En la opción de ***Interact*** se pueden obtener los distintos valores para interactuar con la cosa. Por una parte, se puede utilizar el punto de acceso REST API para actualizar la *shadow* de la cosa. También se pueden obtener los diferentes *topics* para interactuar con la *shadow*.



Desde esta misma consola también es posible acceder a la información almacenada de *shadow* para la cosa creada.



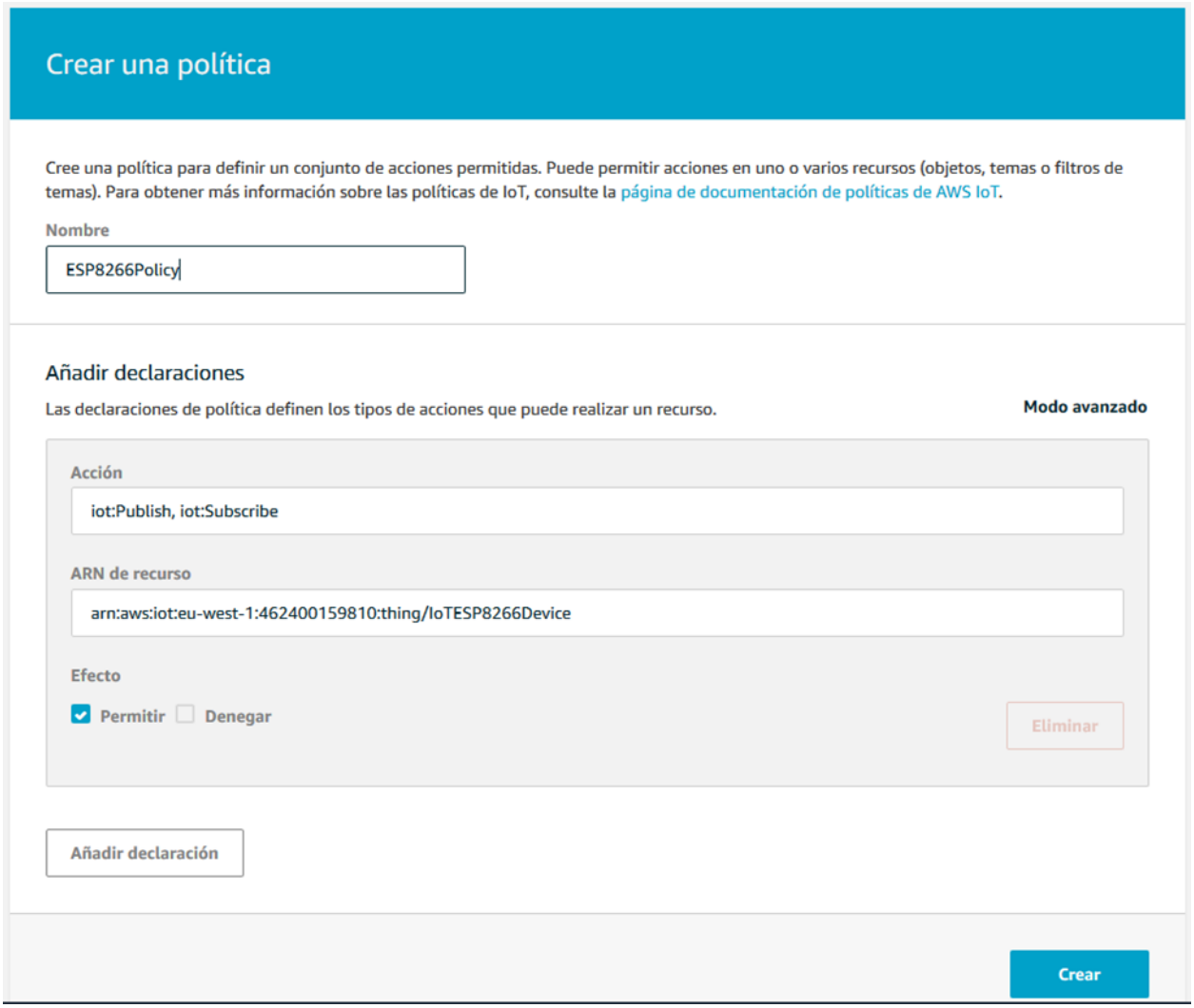
Los certificados X.509 se utilizan para autenticar el dispositivo en AWS IoT Core. Las políticas se utilizan para esta autenticación. Esta política permite realizar operaciones de tipo suscripción o publicación en topics MQTT. Para crear una política de AWS IoT Core hay que realizar los siguientes pasos.

6) Seguro → Políticas → Crear.

7) Añadir decalaraciones: iot:Publicar, iot:Suscribir.

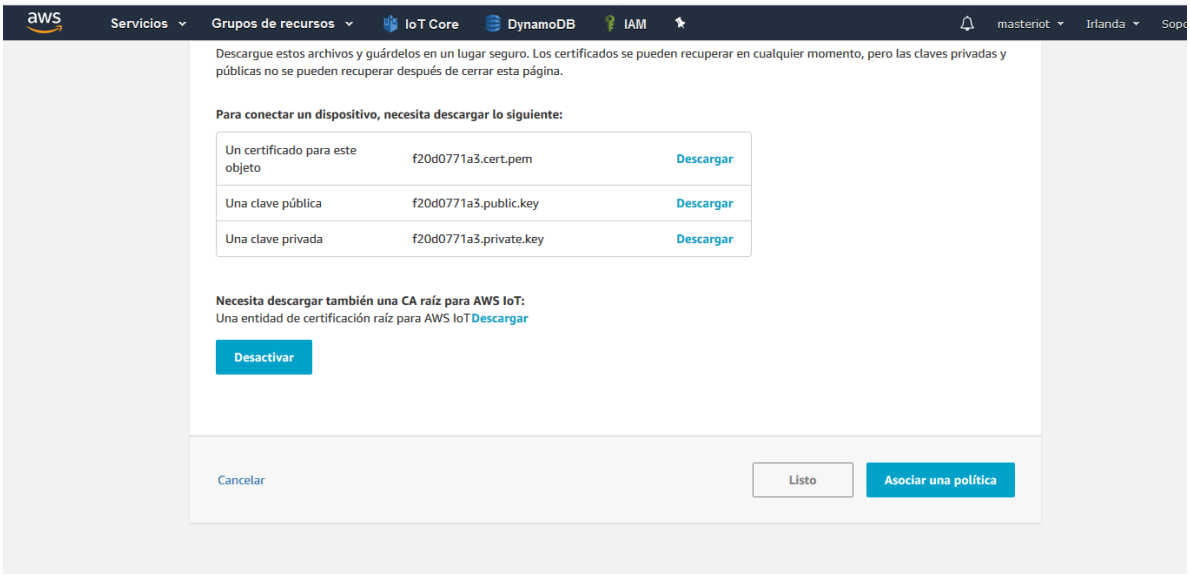
8) ARN del recurso cosa ya creado.

9) Crear.



Gestión de certificados con Arduino IDE

Los certificados que se pueden crear a través de la consola AWS y asociar a la cosa creada son los siguientes.



Los certificados generados durante el proceso de creación de una cosa son los siguientes:

- Certificado para el dispositivo.
- Clave privada.

Adicionalmente necesitaremos descargar el certificado de Amazon: *Amazon CA certificate*

Los ficheros descargados en el ejemplo son:

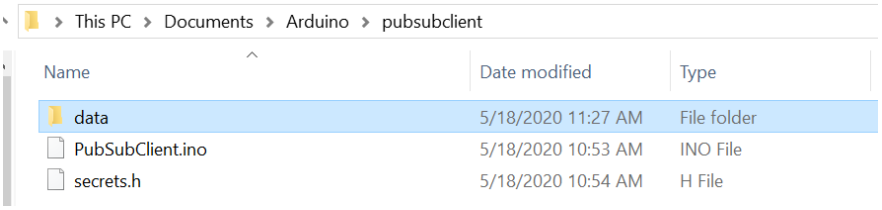
- 48696712d6-certificate.pem.crt
- 48696712d6-private.pem.key
- 48696712d6-public.pem.key
- AmazonRootCA1.pem

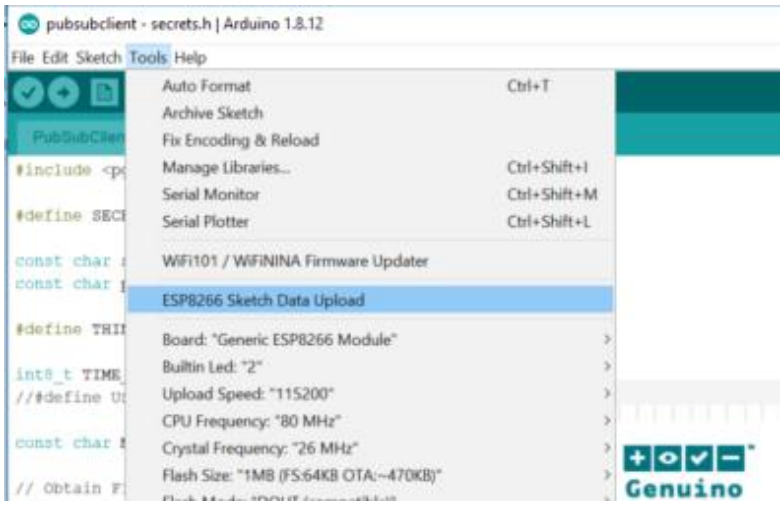
Antes de instalarlos en el dispositivo necesitamos convertirlos al formato DER binario utilizando OpenSSL.

```
openssl x509 -in 48696712d6-certificate.pem.crt -out cert.der -outform DER

openssl rsa -in 48696712d6-private.pem.key -out private.der -outform DER

openssl x509 -in AmazonRootCA1.pem -out ca.der -outform DERconsole.log('Loading function');
```

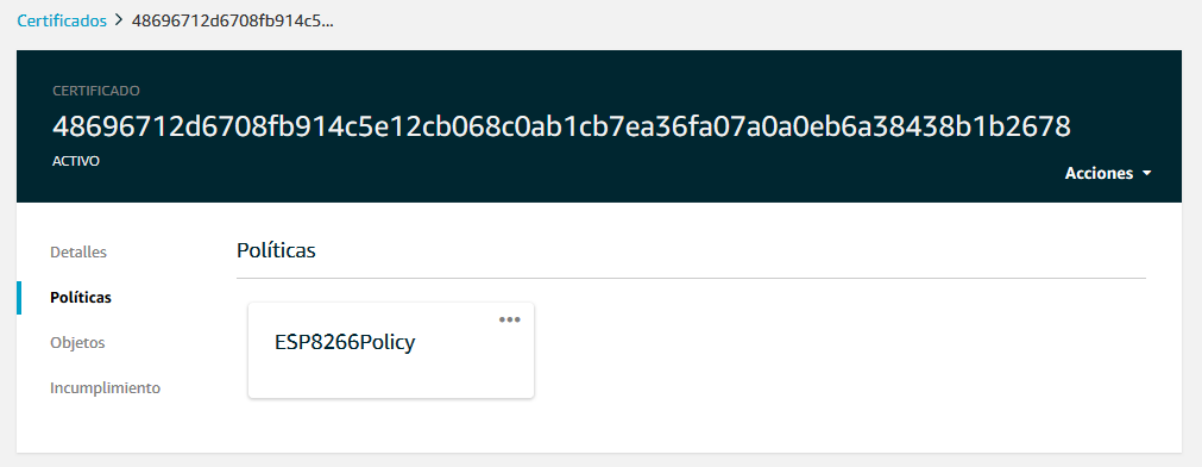
Para poder instalar los certificados en el dispositivo necesitaremos la herramienta ESP8266 Sketch Data Upload en Arduino IDE. Los siguientes ficheros tienen que estar copiados al directorio data del proyecto de Arduino:

- ca.der
- cert.der
- private.der





Es necesario también crear una política asociada al certificado que da permisos al dispositivo para realizar operaciones en AWS IoT.




```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

Creación de usuario

A través del servicio IAM se puede crear los diferentes usuarios necesarios para interactuar con AWS IoT.

1) IAM → Usuarios → Añadir usuario(s).

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	IoTDevicesUser
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The user shown above will be added to the following groups.

Type	Name
Group	IoTDevicesGroup

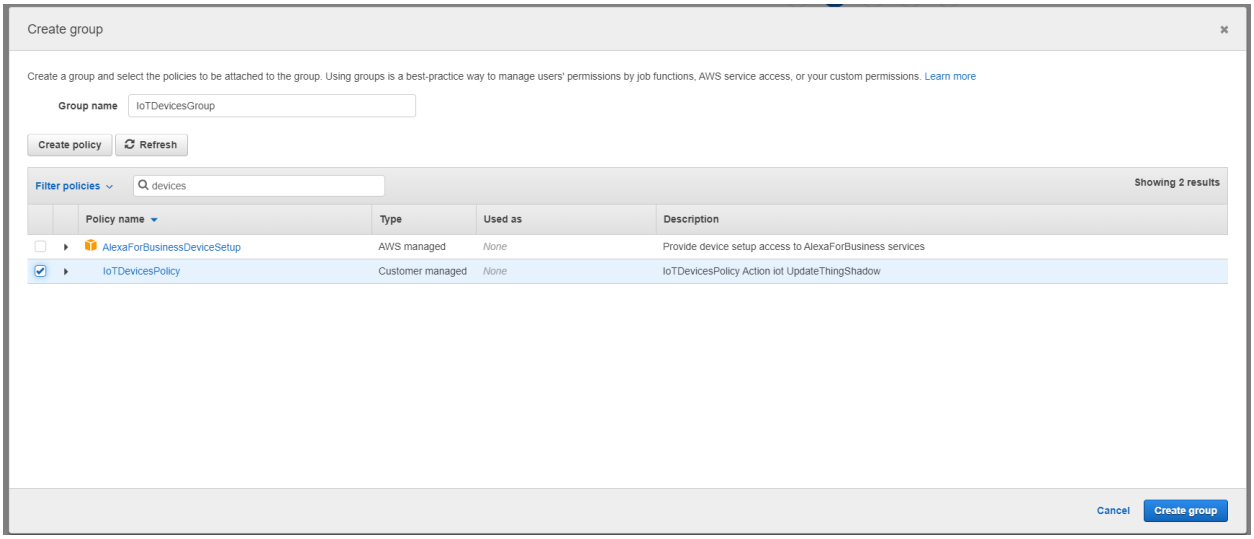
Tags

The new user will receive the following tag

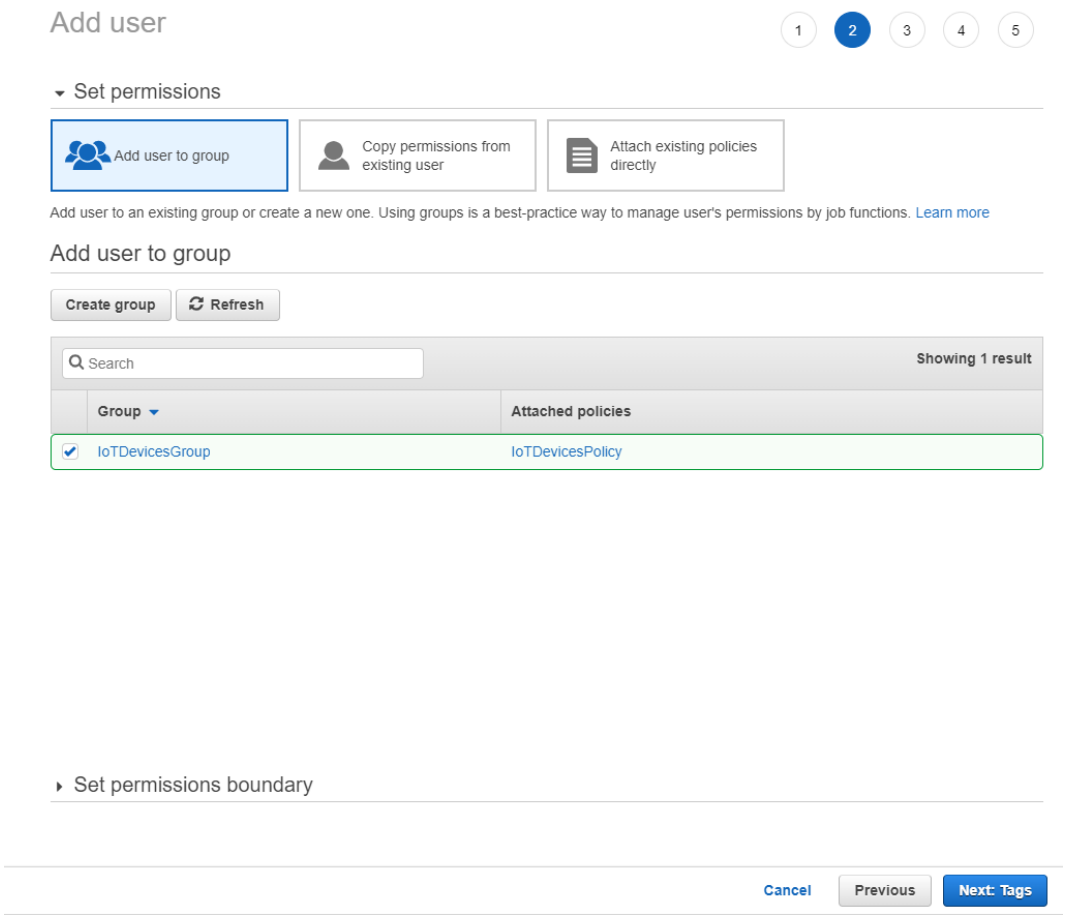
Key	Value
Name	IoTESP8266

Cancel Previous Create user

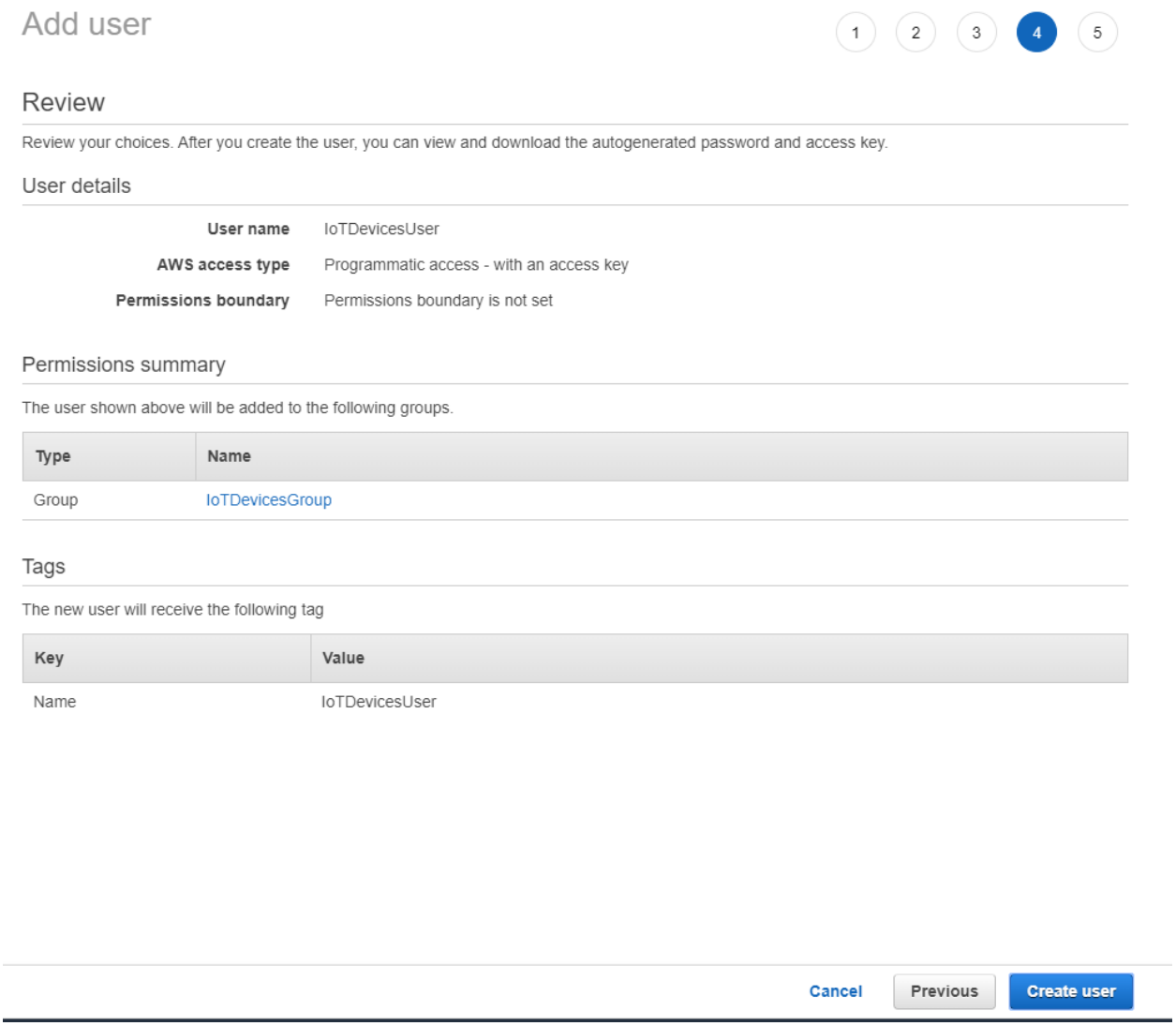
2) Crear Grupo de usuarios.



3) Continuar con la creación del usuario.



4) Continuar con la creación del usuario.



5) Las credenciales que se crean son necesarias para una posible autenticación posterior del dispositivo en AWS IoT. Descargar estas credenciales en formato CSV.

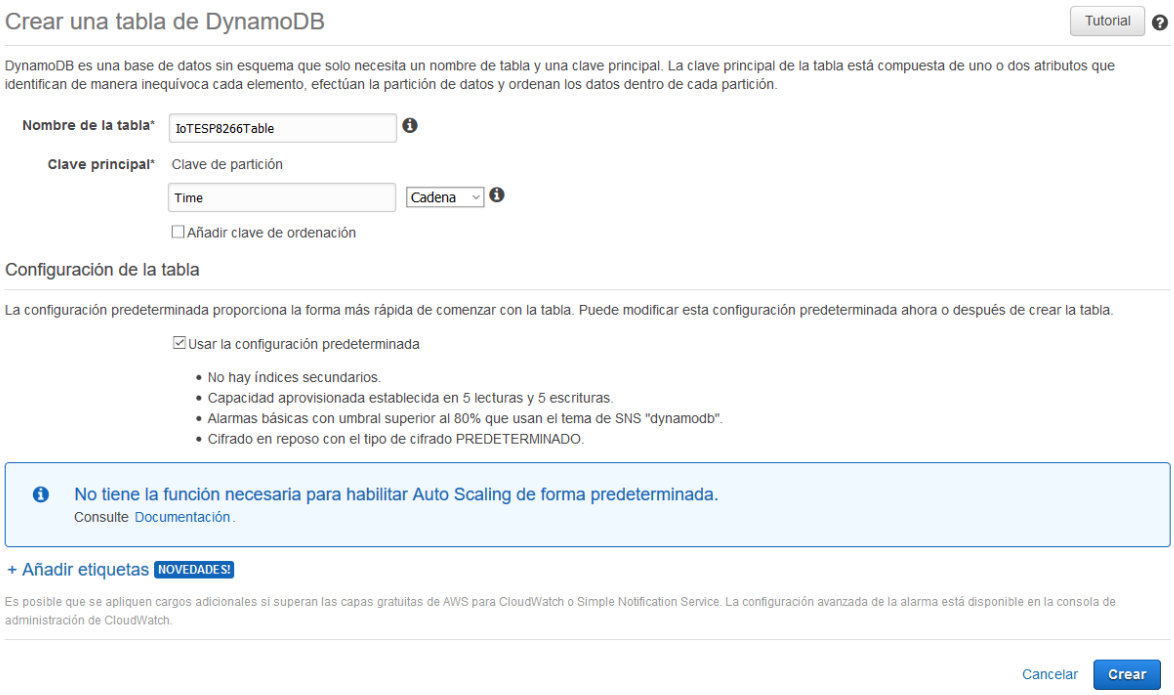
User name	Password	Access key ID	Secret access key	Console login link
IoTDevicesUser		AKIAWX….C	3iZ….	https://accoint_id.signin.aws.amazon.com/console

Creación de tablas en DymamoDB

La información recibida desde el dispositivo se va a almacenar en tablas de DynamoDB para su posible uso posterior. Este es el proceso de creación de una tabla.

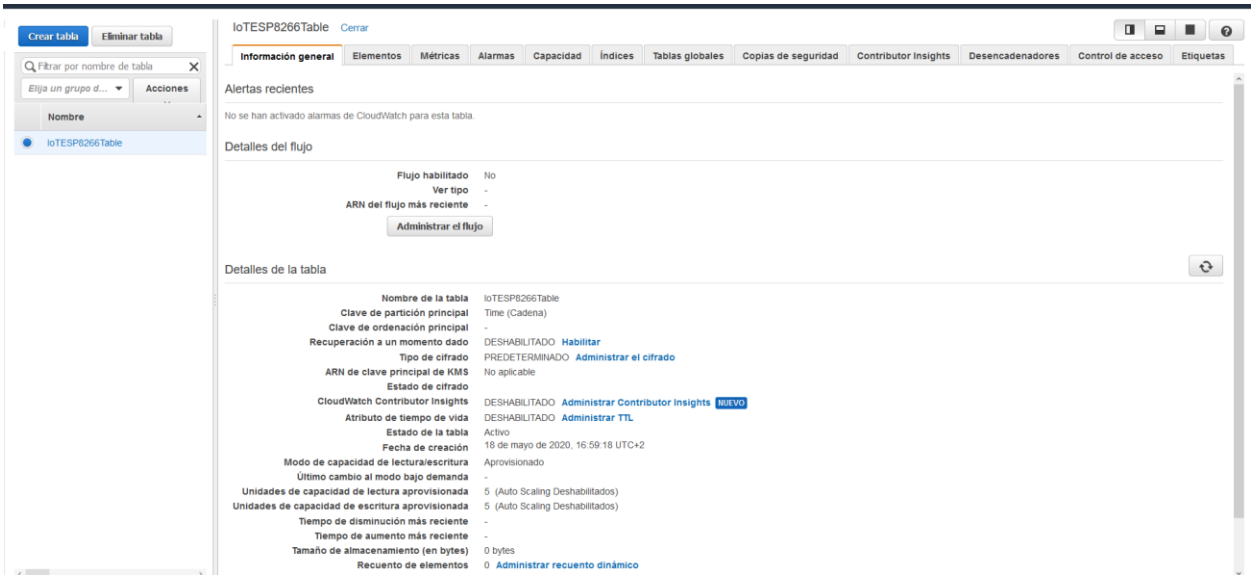
1) DynamoDB → Panel → Crear tabla.

Se identifica la clave principal y se puede añadir de manera opcional una clave de ordenación.

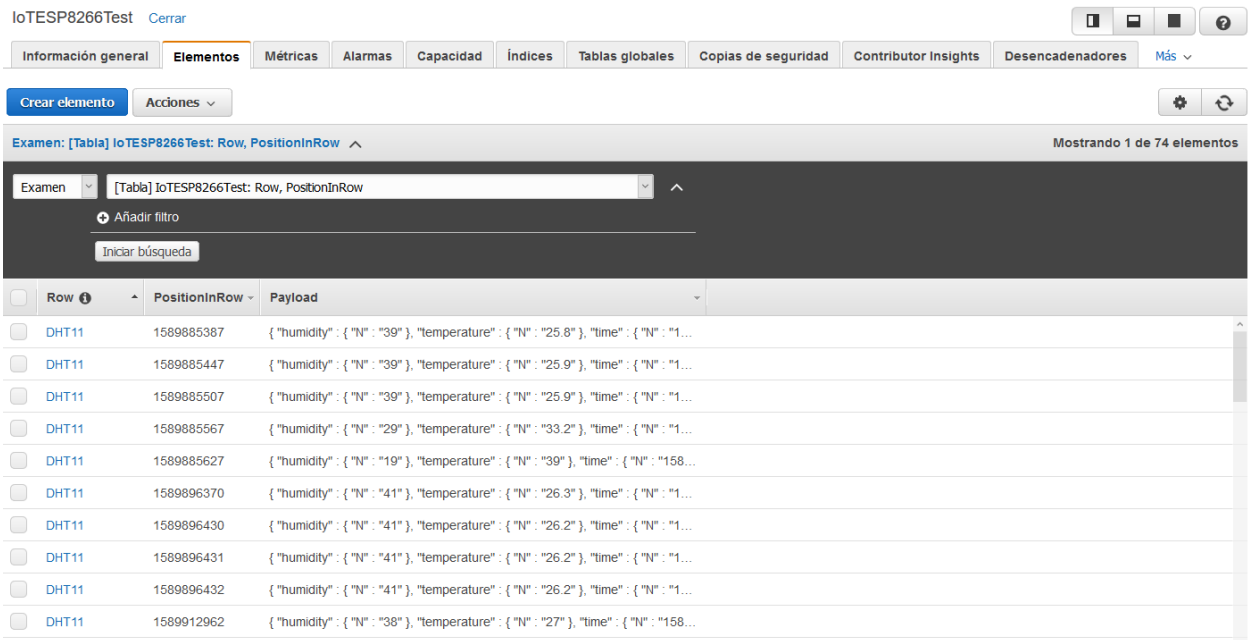


Una vez creada la tabla se puede acceder a su información seleccionándola en el menú.

2) DynamoDB → Tablas → Seleccionar por nombre.

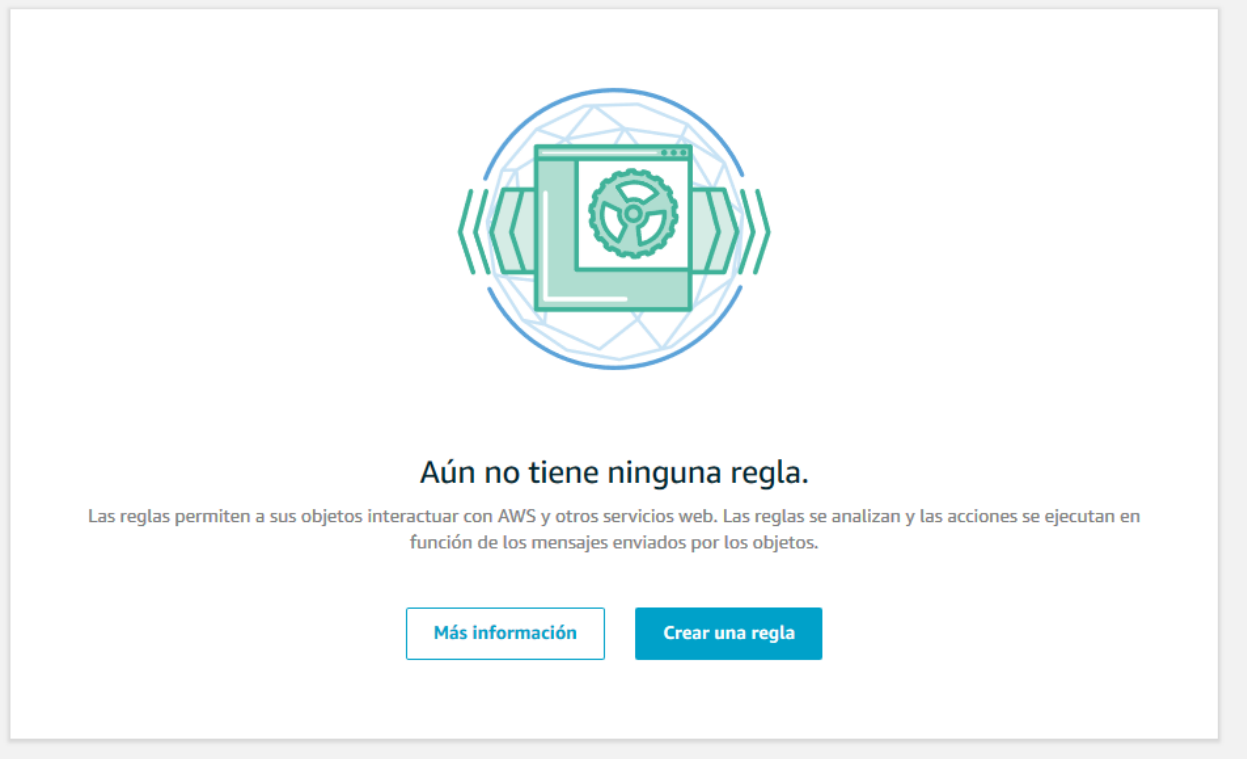


En la pestaña de elementos se puede comprobar el contenido de los elementos de la tabla.

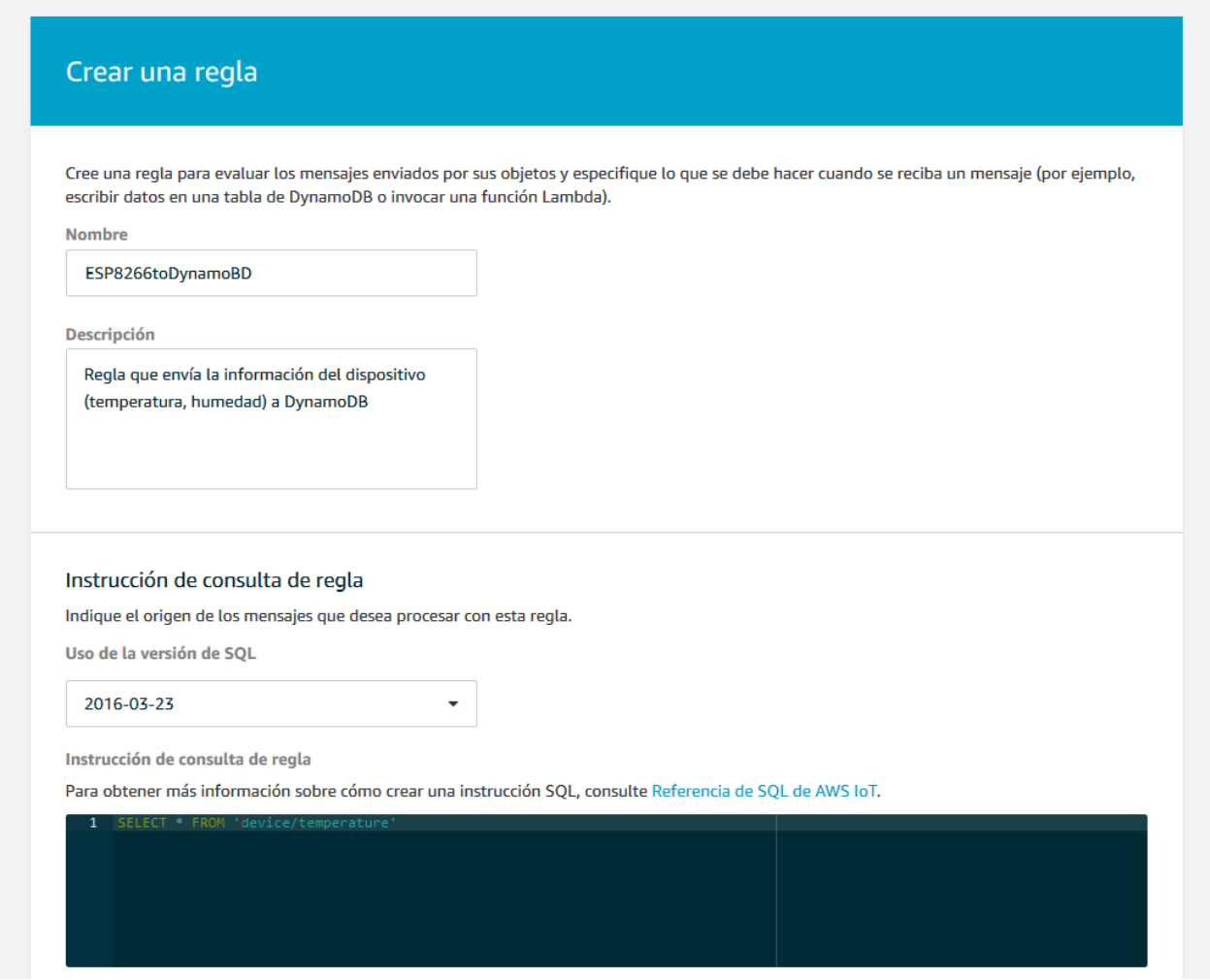


Creación de AWS IoT Rules

1) AWS IoT → Actuar → Reglas→ Crear.

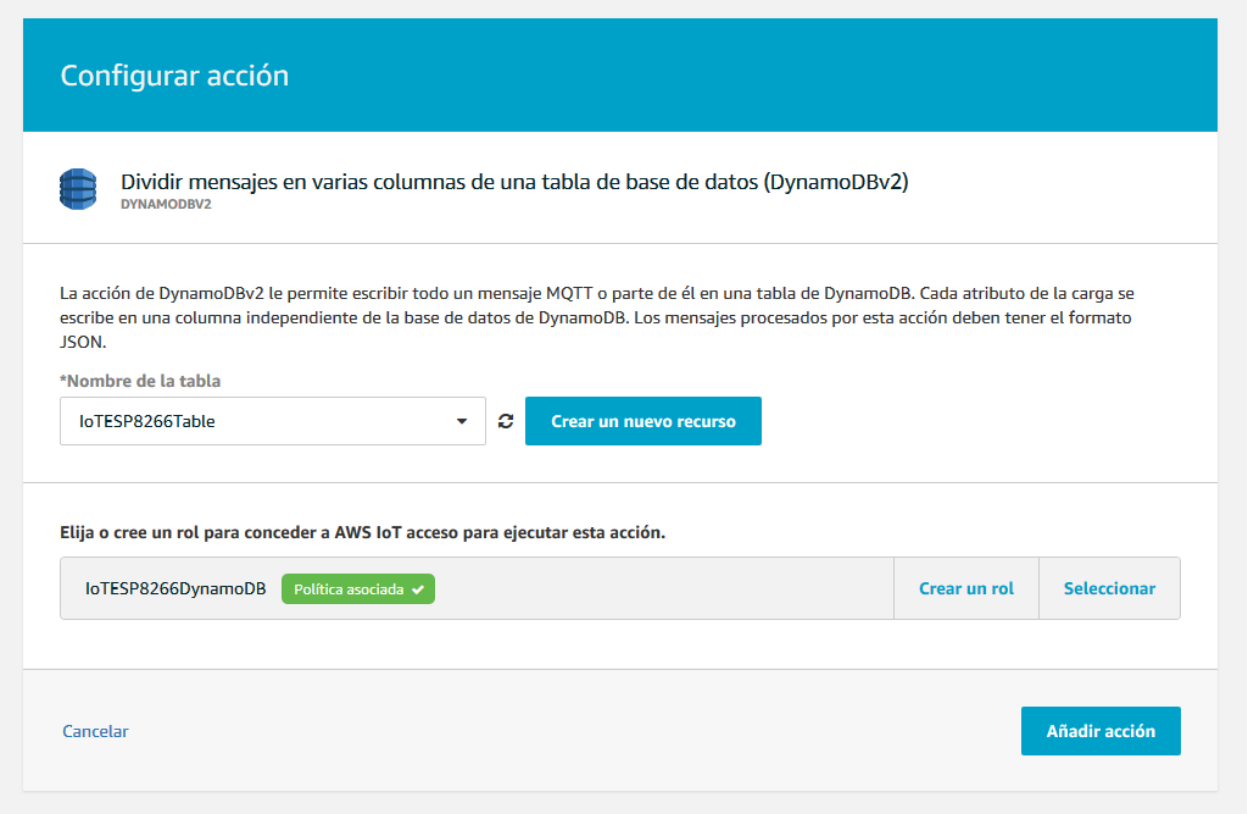


2) Crear una regla.

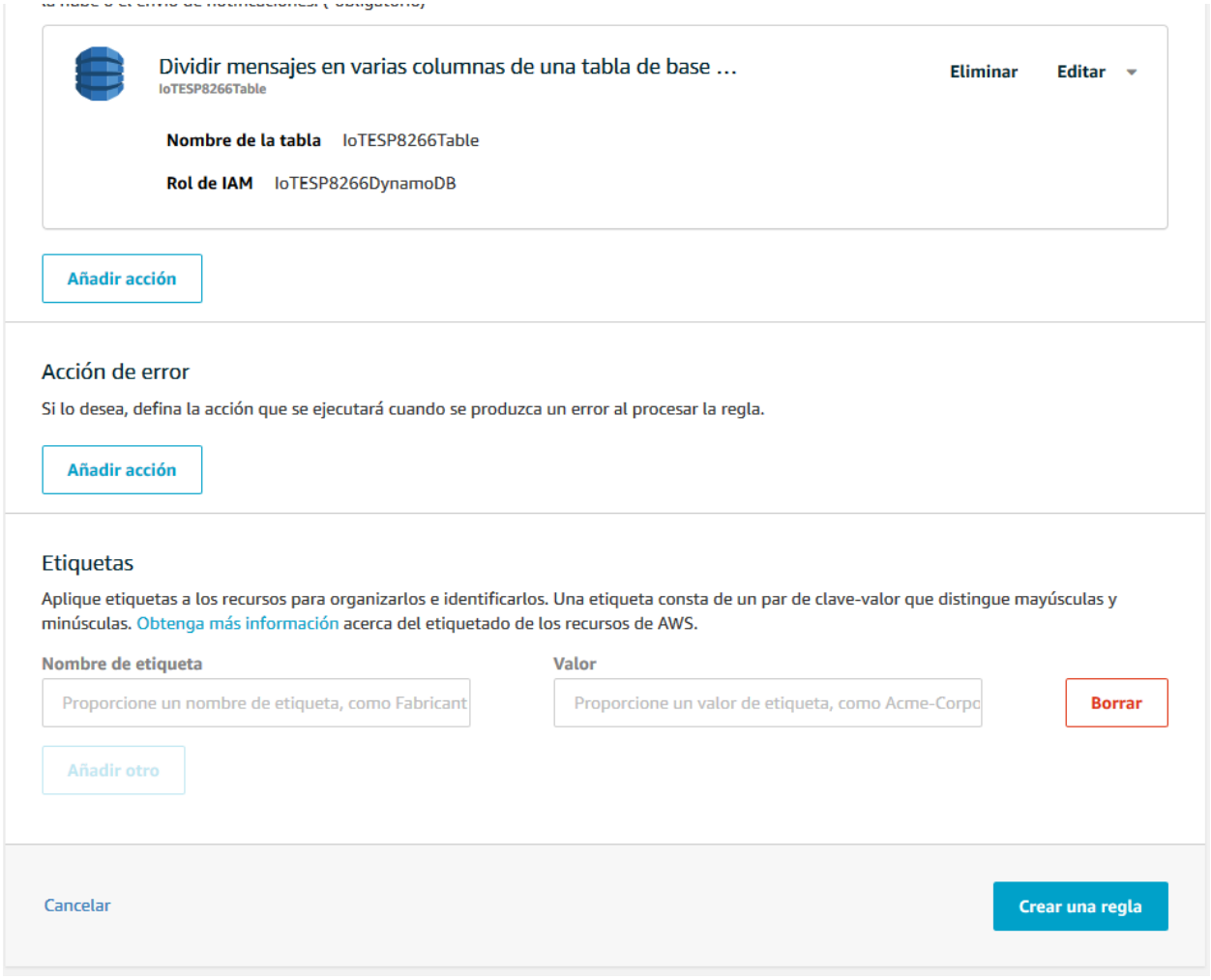


3) Seleccionar acción.

4) Configurar acción.



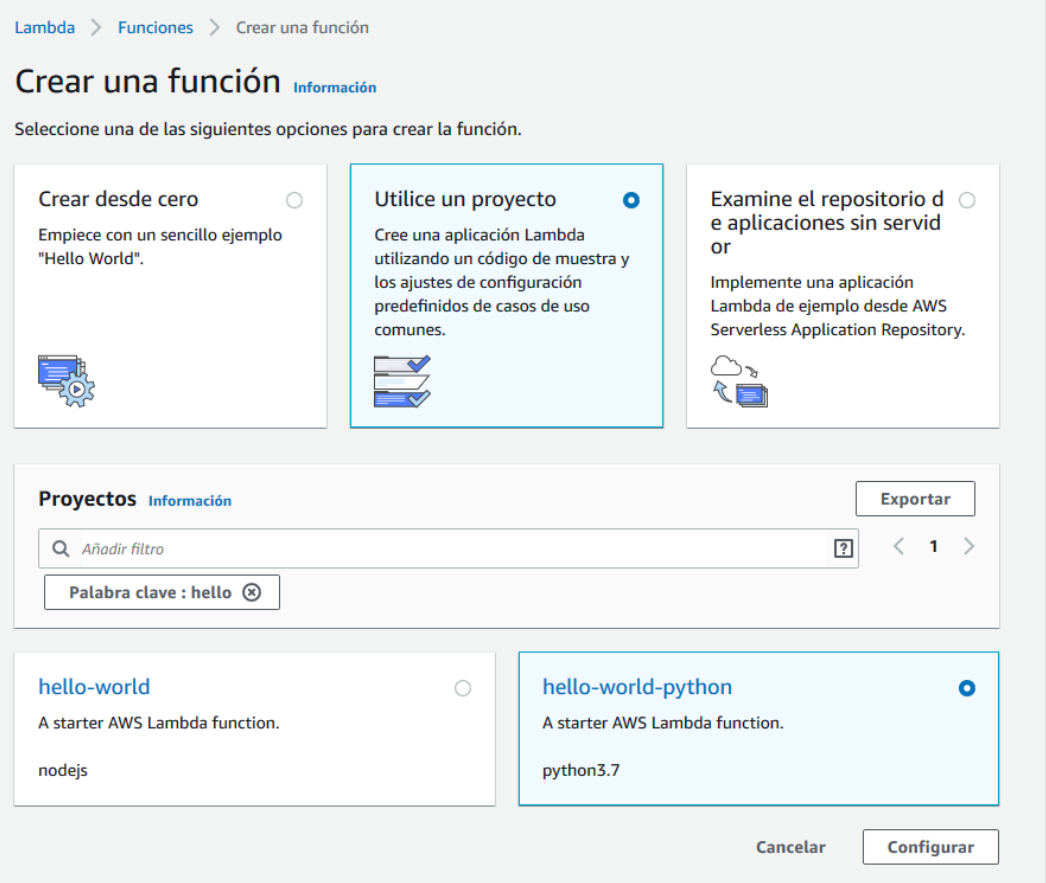
5) Crear regla.



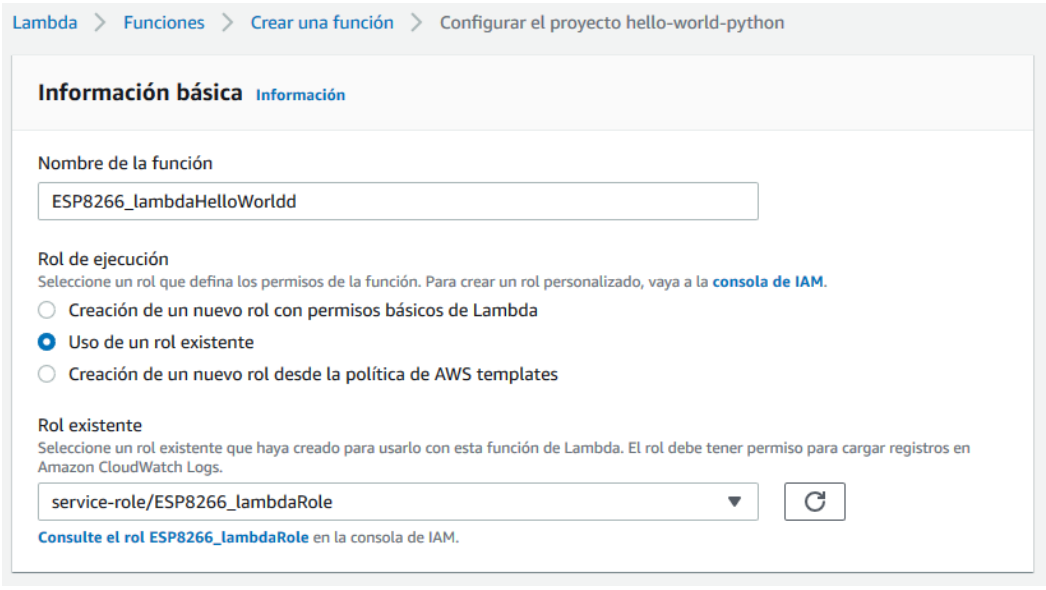
Función Lambda

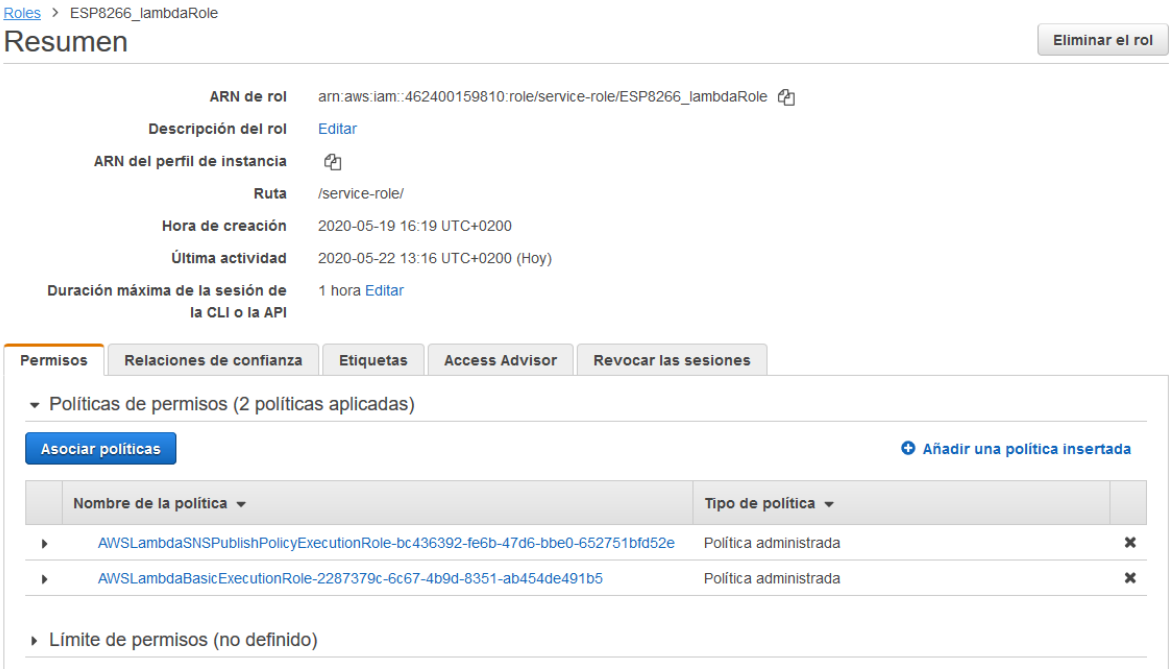
1) AWS Lambda → Funciones → Crear una función.

2) Elegir un proyecto existente. En este caso se utiliza como base el proyecto HelloWorld en código Python3.7.

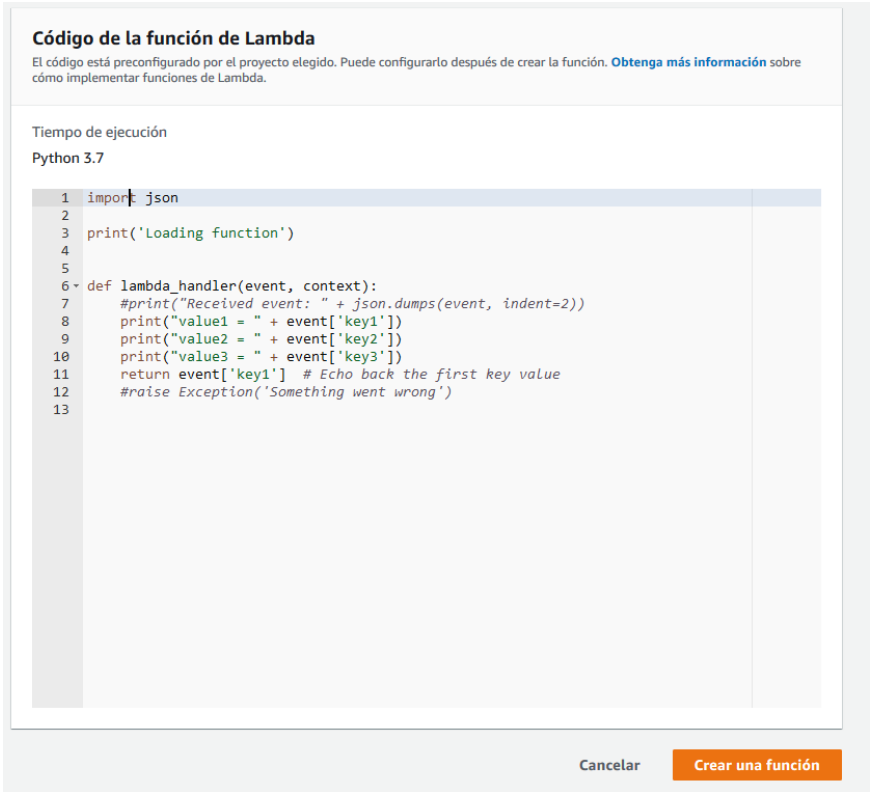


3) Seleccionar un rol para la función lambda.





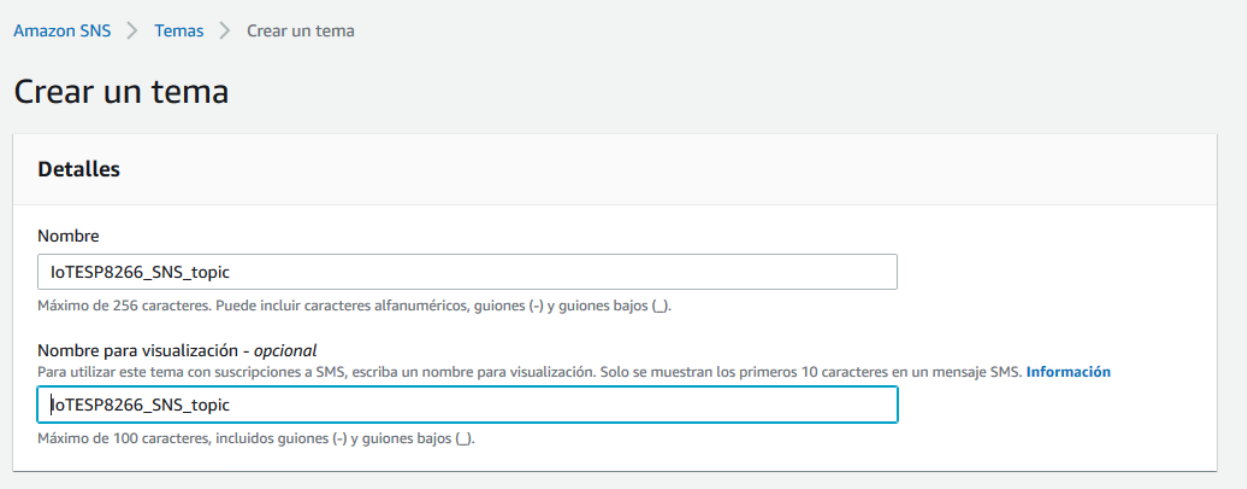
4) Sustituir el código por el de la función que se ha definido para el caso de uso.

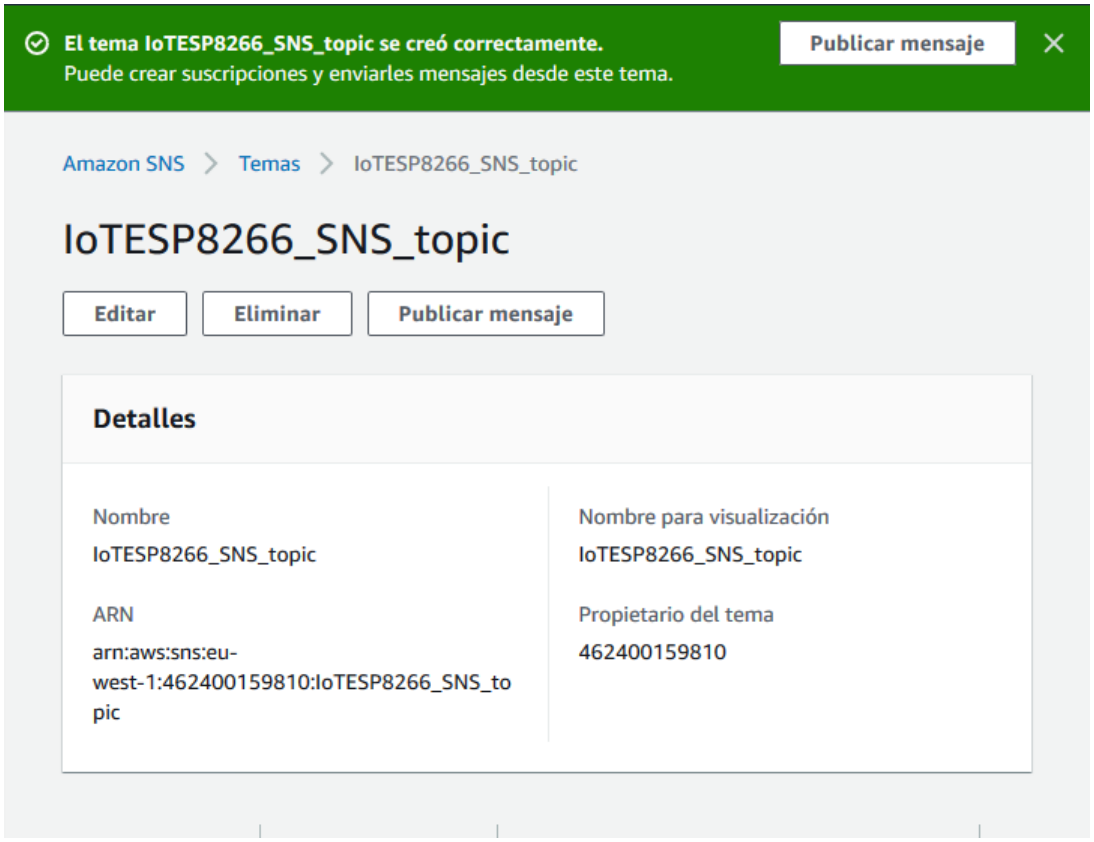


Creación de SNS Topic

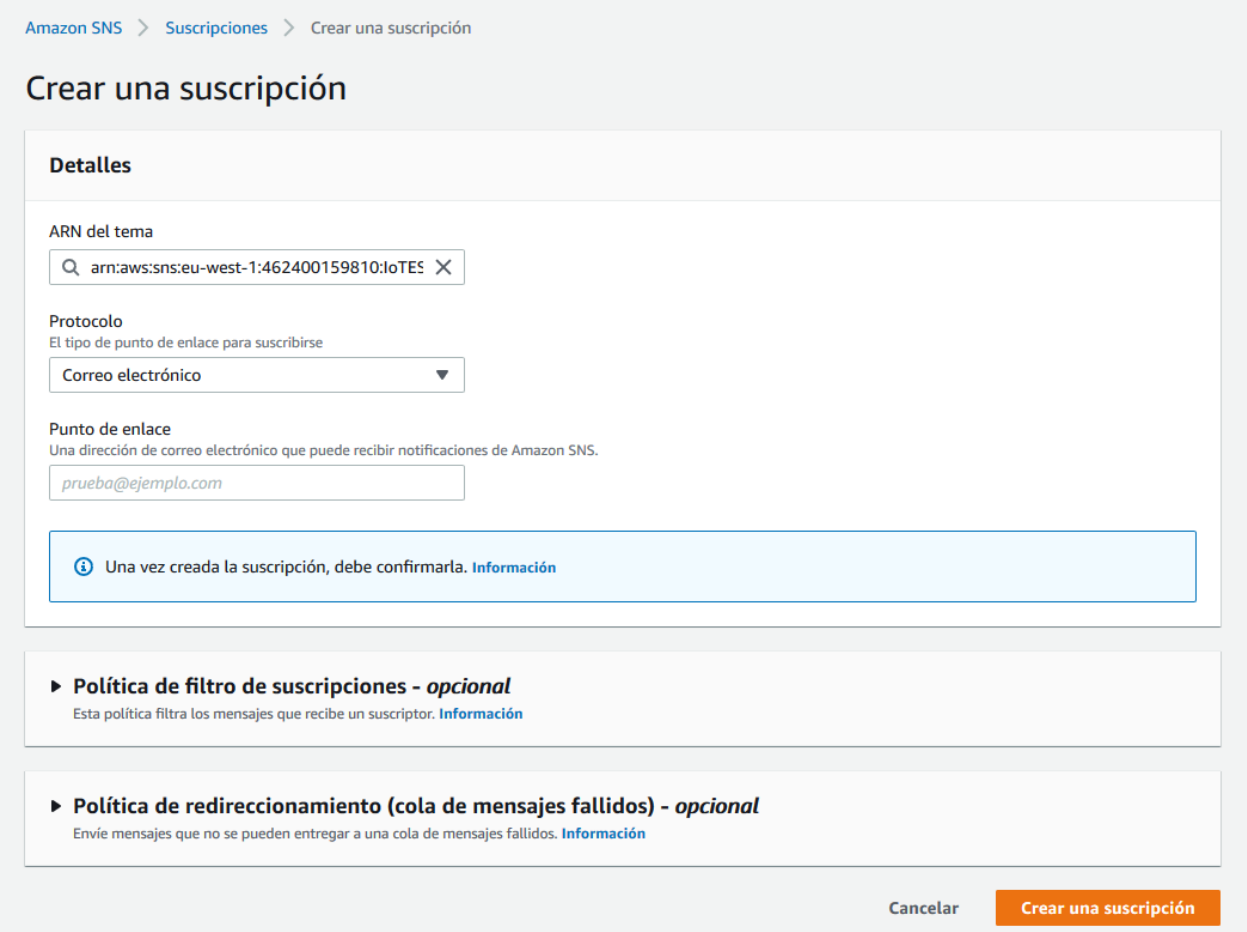
El servicio SNS facilita la entrega de mensajes a puntos de enlace o clientes suscritos. Se va a configurar el envío de mensaje a través de correo.

1) Amazon SNS → Temas → Crear un tema.





2) Crear una suscripción de tipo e-mail.



3) Confirmar la suscripción.

Simple Notification Service

Subscription confirmed!

You have subscribed mipere08@ucm.es to the topic: IoTESP8266_SNS_topic.

Your subscription's id is: `arn:aws:sns:eu-west-1:462400159810:IoTESP8266_SNS_topic:24b9c2ac-903e-48a8-a708-1e55744a214e`

If it was not your intention to subscribe, click here to unsubscribe.

Correo de notificación recibido.

Comprobación de la ejecución correcta de la función Lambda.